STUDY MATERIAL
PROGRAMMING WITH PHP & MYSQL

UNIT - I
INTRODUCTION

PHP

- PHP is a server-side scripting language. It is used to develop Static websites or Dynamic websites or Web applications.
- PHP stands for **Hypertext Pre-processor**, that earlier stood for **Personal Home Pages**.
- PHP scripts can only be interpreted on a server that has PHP installed.
- The client computers accessing the PHP scripts require a web browser only.
- A PHP file contains PHP tags and ends with the extension **".php".**

Commonusesof PHP

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

History of PHP

- The first version of PHP is PHP/FI (Form Interpreter) developed by Ramous Lerdorf, monitoring page view for his online resume.
- This version supports some basic function, capable to handle form data and mSql db.
- PHP/FI 1.0 followed by PHP/FI 2.0 and quickly supplanted in1997 by PHP3.0.
- PHP3.0 developed by Anti Gutmus and Zee Surakshi, complete rewrite of PHP/FI.
- It supports a wide range of database such as MySQL and Oracle.
- In 2003 PHP4.0 was released with better performance, greater reliability, support for web server other than Apache. Support OOPs concept.
- PHP 5.0 support message passing, abstract classes, destructor, better memory management.
- PHP is used on over 15 million website.

Characteristics of PHP

There are many features given by PHP. All Features discussed below one by one.

- Familiarity
- Simplicity
- Efficiency
- Security
- Flexibility
- Open source
- Object Oriented

Familiarity:

If you are in programming background then you can easily understand the PHP syntax. And you can write PHP script because of most of PHP syntax inherited from other languages like C or Pascal.

Simplicity:

PHP provides a lot of pre-define functions to secure your data. It is also compatible with many third-party applications, and PHP can easily integrate with other.

In PHP script there is no need to include libraries like c, special compilation directives like Java, PHP engine starts execution from (<?) escape sequence and end with a closing escape sequence (<?). In PHP script, there is no need to write main function. And also you can work with PHP without creating a class.

Efficiency:

PHP 4.0 introduced resource allocation mechanisms and more pronounced support for object-oriented programming, in addition to session management features. Eliminating unnecessary memory allocation.

Security:

Several trusted data encryption options are supported in PHP's predefined function set. You can use a lot of third-party applications to secure data, allowing for securing application.

Flexibility: -

PHP is a very flexible language because PHP is an embedded language you can embed PHP scripts with HTML, JAVA SCRIPT, WML, XML, and many others. You can run your PHP script on any device like mobile Phone, tabs, laptops, PC and others because of PHP script execute on the server then after sending to the browser of your device.

Free:

PHP is an open source programming language so you can download freely there is no need to buy a licence or anything.

Object Oriented

PHP has added some object-oriented programming features, and Object Oriented programming became possible with PHP 4. With the introduction of PHP 5, the PHP developers have really beefed up the object-oriented features of PHP, resulting in both more speed and added features.

Creating (Declaring) PHP Variables

In PHP, a variable starts with the **$** sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

The variable **$txt** will hold the value **Hello world!** the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, car name, total volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

PHP variable names are case-sensitive.

Output Variables

- The PHP echo statement is often used to output data to the screen.
- The following example will show how to output text and a variable:

Example:

```
<?php
$txt = "Hello World!";
echo "Welcome To $txt!";
?>
```

Output:
Welcome To Hello World!

The following example will produce the same output as the example above:

```
<?php
$txt = "Hello World";
echo "Welcome To " . $txt. "!";
?>
```

Output: Welcome To Hello World!

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.PHP automatically converts the variable to the correct data type, depending on its value.

In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.
The scope of a variable is the part of the script where the variable can be referenced / used.
PHP has three different variable scopes:

- Local
- Global
- Static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example:

```
<?php
$x = 5; // global scope

function myTest() {
   // using x inside this function will generate an error
   echo"<p>Variable x inside function is: $x</p>";
}
myTest();

echo"<p>Variable x outside function is: $x</p>";
?>
```

Output:

Variable x inside function is:
Variable x outside function is: 5

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example:

```
<?php
function myTest() {
   $x = 5; // local scope
   echo"<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an error
echo"<p>Variable x outside function is: $x</p>";
?>
```

Output:

Variable x inside function is: 5
Variable x outside function is:

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

Global Keyword

The global keyword is used to access a global variable from within a function.
To do this, use the global keyword before the variables (inside the function):

Example:

```
<?php
$x = 5;
$y = 10;

functionmyTest(){
   global $x, $y;
  $y = $x + $y;
}
```

```
myTest();
echo $y; // outputs 15
?>
```

Static Keyword

Normally, when a function is completed / executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the static keyword when you first declare the variable:

Example:

```
<?php
function myTest() {
   static $x = 0;
   echo $x;
   $x++;
}

myTest();
myTest();
myTest();
?>
```

Output: 0 1 2

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

Note: The variable is still local to the function.

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

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

Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo"<br>";
echo $y;
?>
```

Output

Hello World
Hello World

PHP Integer

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

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Example:

```
<?php
$x = 5985;
var_dump($x);
?>
```

Output

```
int (5985)
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump () function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

Output:

```
float (10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

Booleans are often used in conditional testing

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump () function returns the data type and value:

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

Output

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

PHP Object

An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared. First we must declare a class of object. For this, we use the class keyword.

A class is a structure that can contain properties and methods:

```
<?php
class Car {
   function Car () {
      $this->model = "VW";
   }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

Output: VW

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.
Variables can also be emptied by setting the value to NULL:

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

Output: NULL

PHP Resource

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

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

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

PHP Arithmetic Operators

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

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression

		on the right
x += y	x = x + y	Addition
x -= y	x = x – y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

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

<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

Increment / Decrement Operators

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

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

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

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

PHP String Operators

PHP has two operators that are specially designed for strings.

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

PHP Array Operators

The PHP array operators are used to compare arrays.

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

Conditional statements are used to perform different actions based on different conditions.

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this. In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...else if ... else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

```
if (condition)

 {
   code to be executed if condition is true;
}
```

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example

```
<?php
$t = date("H");

if ($t <"20") {
   echo"Have a good day!";
}
?>
```

Output: Have a good day!

PHP - The if...else Statement

The if... else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition)

{
   code to be executed if condition is true;
} else {
   code to be executed if condition is false;
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

PHP - The if...elseif ....else Statement

The if....else if ..else statement executes different codes for more than two conditions.

Syntax

```
if (condition)

 {
   code to be executed if this condition is true;
} elseif (condition) {
   code to be executed if this condition is true;
} else {
   code to be executed if all conditions are false;
}
```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example

```
<?php
$t = date("H");

if ($t <"10") {
   echo"Have a good morning!";
} elseif ($t <"20") {
   echo"Have a good day!";
} else {
   echo"Have a good night!";
}
?>
```

The PHP switch Statement

Use the switch statement to select one of the many blocks of code to be executed**.**

Syntax

```
switch (n)

{
   caselabel1:
      code to be executed if n=label1;
      break;
   case label2:
      code to be executed if n=label2;
      break;
   case label3:
```

```
        code to be executed if n=label3;
        break;
Default:
        code to be executed if n is different from all labels;
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example

```
<?php
$favcolor = "red";
switch ($favcolor){
   case “red":
      echo “Your favoritecolor is red!";
     break;
 case"blue":
      echo"Your favoritecolor is blue!";
     break;
 case"green":
      echo"Your favoritecolor is green!";
     break;
 default:
      echo"Your favoritecolor is neither red, blue, nor green!";
}
?>
```

Output: Your favourite color is red!

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- for each - loops through a block of code for each element in an array

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

while (*condition is true*)

```
{
  code to be executed;
}
```

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):

Example

```
<?php
$x = 1;
while($x <= 5) {
  echo"The number is: $x <br>";
  $x++;
}
?>
```

Output :

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

do

```
{
  code to be executed;
} while (condition is true);
```

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;
do {
   echo"The number is: $x <br>";
   $x++;
} while ($x <= 5);
?>
```

Output:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

UNIT – II
ARRAYS AND FUNCTIONS

Arrays

An array stores multiple values in one single variable:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

Create an Array in PHP

In PHP, the array() function is used to create an array:

array();

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:
The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

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

```
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.
There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

The named keys can then be used in a script:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Sort Functions For Arrays

we will go through the following PHP array sort functions:

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort()- sort associative arrays in descending order, according to the value
- krsort()- sort associative arrays in descending order, according to the key

Sort Array in Ascending Order - sort()
The following example sorts the elements of the $cars array in ascending alphabetical order:

Example

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

The following example sorts the elements of the $numbers array in ascending numerical order:

Example

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

Sort Array in Descending Order - rsort()

The following example sorts the elements of the $cars array in descending alphabetical order:

Example

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

The following example sorts the elements of the $numbers array in descending numerical order:

Example

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

Sort Array (Ascending Order), According to Value - asort()

The following example sorts an associative array in ascending order, according to the value:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

Sort Array (Ascending Order), According to Key - ksort()

The following example sorts an associative array in ascending order, according to the key:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

Sort Array (Descending Order), According to Value - arsort()

The following example sorts an associative array in descending order, according to the value:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

Sort Array (Descending Order), According to Key - krsort ()

The following example sorts an associative array in descending order, according to the key:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);
?>
```

UNIT – III
FILE HANDLING

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

PHP readfile() Function

The readfile() function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Pre-processor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensibleMarkup Language

The PHP code to read the file and write it to the output buffer is as follows (the readfile () function returns the number of bytes read on success)

Example

```
<?php
echo readfile("webdictionary.txt")
?>
```

PHP Open File - fopen ()

A better method to open files is with the fopen () function. This function gives you more options than the readfile () function.

We will use the text file, "webdictionary.txt", during the lessons:

The first parameter of fopen () contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen () function is unable to open the specified file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

The file may be opened in one of the following modes:

Modes	Description
R	**Open a file for read only**. File pointer starts at the beginning of the file
W	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

A	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
X	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread ()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

PHP Close File - fclose ()

The **fclose ()** function is used to close an open file.

The **fclose** () requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
$myfile = fopen ("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

PHP Read Single Line - fgets ()

The fgets() function is used to read a single line from a file.

The example below outputs the first line of the "webdictionary.txt" file:

Example

```
<?php
$myfile = fopen ("webdictionary.txt", "r") or die ("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

PHP Create File - fopen ()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use fopen () on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```
$myfile = fopen ("testfile.txt", "w")
```

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - fwrite ()

The fwrite () function is used to write to a file.

The first parameter of fwrite () contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose () function.

If we open the "newfile.txt" file it would look like this:
John Doe
Jane Doe

Cookie

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the set cookie () function.

Syntax

setcookie (*name, value, expire, path, domain, secure, httponly*);
Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset()function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
   echo "Cookie named '" . $cookie_name. "' is not set!";
} else {
   echo "Cookie '" . $cookie_name. "' is set!<br>";
   echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Note: The setcookie () function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URL encoded when sending the cookie, and automatically decoded when received (to prevent URL encoding, use setrawcookie () instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
   echo "Cookie named '" . $cookie_name. "' is not set!";
} else {
   echo "Cookie '“. $cookie_name. "' is set! <br>";
   echo "Value is: “. $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie () function with an expiration date in the past:

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie () function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>
<?php
```

```
if(count($_COOKIE) > 0) {
   echo "Cookies are enabled.";
} else {
   echo "Cookies are disabled.";
}
?>
</body>
</html>
```

Session

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer.

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favoritecolor, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a **database.**

Start a PHP Session

A session is started with the session_start () function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
```

```
echo "Session variables are set.";
?>
</body>
</html>
```

Note: The session_start () function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start ()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start ();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favoritecolor is “. $_SESSION["favcolor"]. ”. <br>";
echo "Favorite animal is “. $_SESSION["favanimal"]. ".";
?>
</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start ();
?>
<!DOCTYPE html>
<html>
<body>
<?php
print_r($_SESSION);
?>
</body>
</html>
```

UNIT – IV
EFFECTIVENESS OF MYSQL

What is a Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as **Foreign Keys**.

A **Relational DataBase Management System (RDBMS)** is a software that −

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

Database − A database is a collection of tables, with related data.

Table − A table is a matrix with data. A table in a database looks like a simple spreadsheet.

Column − One column (data element) contains data of one and the same kind, for example the column postcode.

Row − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.

Redundancy − Storing data twice, redundantly to make the system faster.

Primary Key − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.

Foreign Key − A foreign key is the linking pin between two tables.

Compound Key − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.

Index − An index in a database resembles an index at the back of a book.

Referential Integrity − Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

MySQL uses many different data types broken into three categories −

- Numeric
- Date and Time
- String Types.

Numeric Data Types

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you.

The following list shows the common numeric data types and their descriptions −

INT

A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.

TINYINT

A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.

SMALLINT

A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.

MEDIUMINT

A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.

BIGINT

A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.

FLOAT(M,D)

A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

DOUBLE(M,D)

A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

DECIMAL(M,D)

An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types

The MySQL date and time data types are as follows −

DATE

A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.

DATE TIME

A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.

TIME STAMP

A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).

TIME

Stores the time in a HH:MM:SS format.

YEAR(M)

Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

String Types

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

CHAR(M)

A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.

VARCHAR(M)

A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.

BLOB or TEXT

A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are **case sensitive** on BLOBs and are **not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.

TINYBLOB or TINYTEXT

A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.

MEDIUMBLOB or MEDIUMTEXT

A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.

LONGBLOB or LONGTEXT

A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.

ENUM

An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

To begin with, the table creation command requires the following details −

- Name of the table
- Name of the fields
- Definitions for each field

Syntax

Here is a generic SQL syntax to create a MySQL table −

```
CREATE TABLE table_name (column_namecolumn_type);
```

Now, we will create the following table in the **TUTORIALS** database.

```
create table tutorials_tbl (

tutorial_id INT NOT NULL AUTO_INCREMENT,

tutorial_title VARCHAR(100) NOT NULL,

tutorial_author VARCHAR (40) NOT NULL,

submission_date DATE,

  PRIMARY KEY (tutorial_id)

);
```

Here, a few items need explanation −

- Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error.
- Field Attribute **AUTO_INCREMENT** tells MySQL to go ahead and add the next available number to the id field.
- Keyword **PRIMARY KEY** is used to define a column as a primary key. You can use multiple columns separated by a comma to define a primary key.

UNIT - V
PHP WITH MYSQL

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

Create a MySQL Database Using MySQLi

The CREATE DATABASE statement is used to create a database in MySQL.

Create a MySQL Table Using MySQLi

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

Insert Data into MySQL Using MySQLi

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Now, let us fill the table with data.

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

The following examples add a new record to the "MyGuests" table:

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
if (mysqli_query($conn, $sql)) {
   echo "New record created successfully";
} else {
   echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Select Data from a MySQL Database

The SELECT statement is used to select data from one or more tables:
SELECT column_name(s) FROM table_name
or we can use the * character to select ALL columns from a table:
SELECT * FROM table_name

Select Data WithMySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
```

```
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);
if ($result->num_rows> 0) {
   // output data of each row
   while($row = $result->fetch_assoc()) {
      echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
   }
} else {
   echo "0 results";
}
$conn->close();
?>
```

Output:

id: 1 - Name: John Doe
id: 2 - Name: Mary Moe
id: 3 - Name: Julie Dooley

Delete Data From a MySQL Table Using MySQLi

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Notice the WHERE clause in the DELETE syntax:

The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

Example (MySQLi)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli ($servername, $username, $password, $dbname);
```

```
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";
if ($conn->query($sql) === TRUE) {
echo "Record deleted successfully";
} else {
echo "Error deleting record: " . $conn->error;
}
$conn->close();
?>
```