PHP

PHP is a server side scripting language 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.

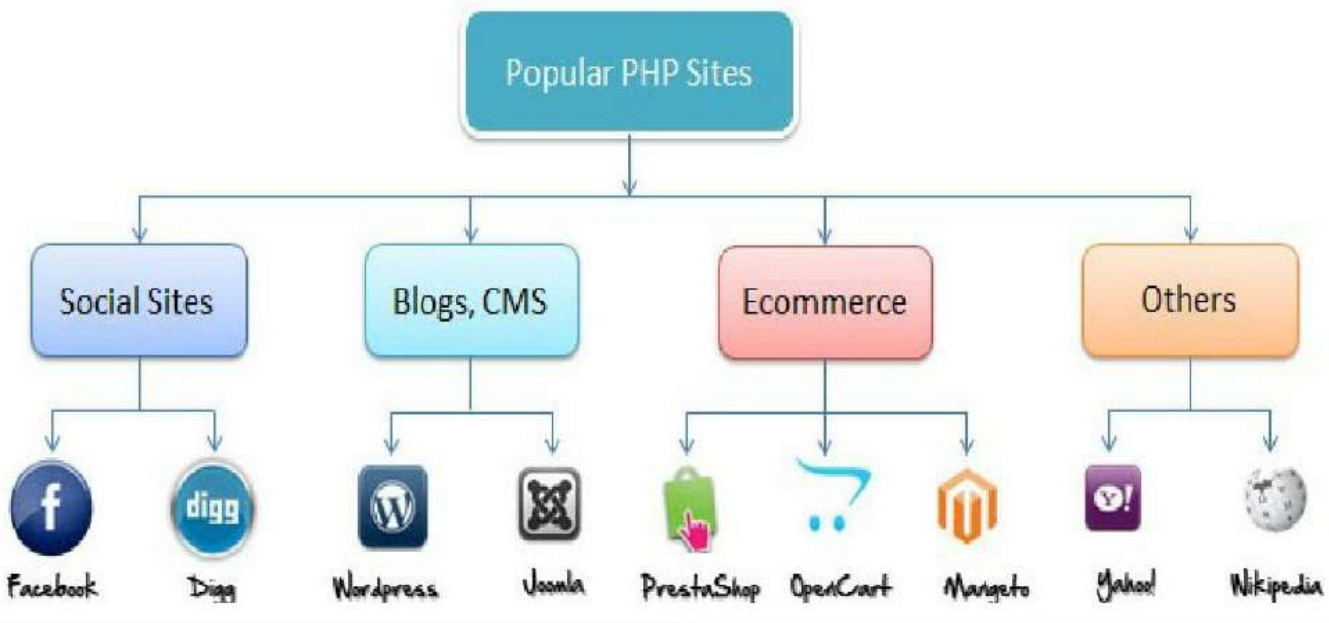
The client computers accessing the PHP scripts require a web browser only.

PHP code may be embedded into HTML code.

A PHP file contains PHP tags and ends with the extension ".php".

Why use PHP?

- PHP is **open source and free.**
- Short learning curve compared to other languages such as JSP, ASP etc.
- Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost effective choice.
- PHP is regular updated to keep abreast with the latest technology trends.
- Other benefit that you get with PHP is that it's a **server side scripting language**; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is **cross platform;** this means you can deploy your application on a number of different operating systems such as windows, Linux, Mac OS etc.



Php data types:

There is no specific datatype in php. We can assign any datatype to the variable through var keyword.

Int datatype:

var a=10;

float datatype:

var a=10.53;

char datatype:

var a='m';

String datatype:

var a="muni";

Example:

```
<?php
$a=5;
$b=150.155;
$c='s';
$d="IV SEM MCA";
echo "Integer value is".$a."<br/>";
echo "Float value is".$b."<br/>";
echo "Single character is".$c."<br/>";
echo "String is".$d;
?>
```

Line Break; Used to give a line break

```
echo "variable or string <br/>"
```

Addition of two numbers

```
<?php
$n1=10;
$n2=20;
```

```
$sum=$n1+$n2;

echo "addition is $sum";

?>
```

Control Structures

The control structures within Javascript allow the program flow to change within a unit of code or function. These statements can determine whether or not given statements are executed, as well as repeated execution of a block of code.

Contents

1. If...Else Statements

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

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

(i) If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

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

Example

```
<?php
    $a=10;
    $b=20;
    if($a<$b)
    echo "b is big";
?>
```

(ii) If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
        code to be executed if condition is true
  }
else
  {
        code to be executed if condition is not true
  }
```

Example

```
<?php
        $a=25;
        $b=20;
        if($a<$b)
        echo "b is big";
        else
        echo "a is big";
?>
```

(iii) If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
  {
        code to be executed if condition1 is true
  }
else if (condition2)
  {
        code to be executed if condition2 is true
  }
else
  {
        code to be executed if  the condition1 and condition2 is FALSE
  }
```

Example

```
<?php
         $x=10;
         $y=5;
        if($x==$y)
        {
        echo "Condition1 true <br/>";
        }
        else if($x<=$y)
        {
        echo "Condition2 true <br/>";
        }
        else
        {
```

```
        echo "Condition 3 true";
        }
?>
```

2. Loops

- Loops execute a block of code a specified number of times, or while a specified condition is true.
- 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 lines in a script we can use loops to perform a task like this.

(i) While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
intialization
while (condition)
{
code to be executed
increment or decrement
}
```

Example

```
<?php
echo "Numbers are <br/>";
$a=1;
while($a<=5)
{
 echo "$a <br/>";
 $a++;
}
?>
```

Output

Numbers are
1
2
3
4
5

(ii). do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
intialization
```

```
do
{
        code to be executed
        increment or decrement
}
while (condition);
```

Example

```
<?php
        echo "Numbers are <br/>";
        $a=1;
        do
        {
         echo "$a <br/>";
         $a++;
        }while($a<=5);
?>
```

Numbers are
1
2
3
4
5

(ii) The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (intialization;condition;increment or decrement)
{
        code to be executed
}
```

Example

```
<?php
echo "Numbers are <br/>";
for($a=1;$a<=5;$a++)
{
 echo "$a <br/>";
}
?>
```

Numbers are
1
2
3
4
5

Switch Statement

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

Syntax

```
switch(variablename)
{
        case 1:
                execute code block 1
                break;
        case 2:
                execute code block 2
```

```
        break;
    default:
        code to be executed if variable value  is different
        from case 1 and 2
}
```

Example

```
<?php
    $a=1;
    switch($a)
    {
      case 1:echo "mega star";break;
      case 1:echo "power star";break;
      case 1:echo "super star";break;
    }
?>
```

Output

mega star

Php Programs

Example 1:Even or Odd Program

```
<?php
$number=1233456;
if($number%2==0)
{
 echo "$number is Even Number";
}
else
{
 echo "$number is Odd Number";
}
?>
```

Example 2: Factorial Number

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

Example 3: Palindrome number Program in PHP

```
<?php
$num = 121;
$p=$num;
$revnum = 0;
while ($num != 0)
```

```
{
$revnum = $revnum * 10 + $num % 10;
//below cast is essential to round remainder towards zero
$num = (int)($num / 10);
}
if($revnum==$p)
{
echo $p," is Palindrome number";
}
else
{
echo $p." is not Palindrome number";
}
?>
```

Example 4: Fibonacci Series

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

Arrays

An array stores multiple values in one single variable:

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

1. PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

$arrayname = array(element1,element2, element3);

$arrayname = array("element1","element2", "element3"); //string

or the index can be assigned manually:

$arrayname[0] = element1;
$arrayname[1] = element2;
$arrayname [2] = element3;

Printing elements

echo arrayname[0].arrayname[1].arrayname[2].arrayname[3];

Example:

```
<?php
$a=array(5,15,4,12);
echo "Array elements are";
echo $a[0]."<br/>";
echo $a[1]."<br/>";
echo $a[2]."<br/>";
echo $a[3];
?>
```

Output

Array elements are
5
15
4
12

Get Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

```
$a=array(5,15,4,12,8,7,14);
echo count($a);
```

Output

7

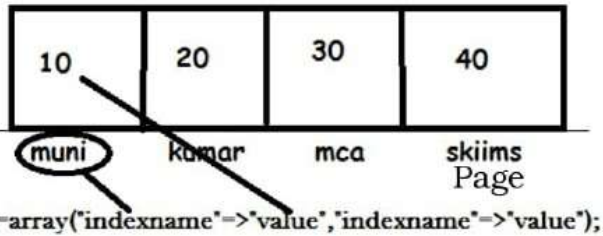
2. PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

$a=array("muni"=>"25","kumar"=>"15","hema"=>"21");

or:



$a['muni']= “25";
$a['kumar']= “15";
$ a['hema'] = “21";

The named keys can then be used in a script:

Example

```
<?php
$a=array("muni"=>"10","kumar"=>"20","mca"=>"30","skiims"=>"40");
echo "Array elements are"."<br/>";
echo $a['muni']."<br/>";
echo $a['kumar']."<br/>";
echo $a['mca'] ."<br/>";
echo $a['skiims'];
?>
```

3. Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

```
$arrayname = array(

        array(element1,element2,element3), //First row with row index
position 0
        array(element1,element2,element3) //Second row with row index
position 1
         );
```

Example

```
<?php
$a=array(array(10,20,30),array(40,50,60));
echo $a[0][0];
echo $a[0][1];
echo $a[0][2];
echo "<br/>";
echo $a[1][0];
echo $a[1][1];
echo $a[1][2];
?>
```

	0	1	2
0	10	20	30
1	40	50	60

Sorting Arrays

1. Sort Array in Ascending Order - sort()

sort(). This is a predefined method used to sort the array elements in ascending order.

Syntax: sort($a);

The following example sorts the elements of the $a array in ascending alphabetical order:

Example

$a=array(25,14,18,3,19,17,13,10);

sort($a)

```
<?php
$a=array(25,14,18,3,19,17,13,10);
$length=count($a);
echo "Before sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
 echo $a[$i]."<br/>";
}
sort($a);
echo "After sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
 echo $a[$i]."<br/>";
}
?>
```

Before sorting
25
14
18
3
19
17
13
10
After sorting
3
10
13
14
17
18
19
25

Before sorting
25
14
18
3
19
17
13
10
After sorting
3
10
13
14
17
18
19
25

2. Sort Array in Descending Order - rsort()

rsort(). This is a predefined method used to sort the array elements in descending order.

Syntax: rsort($a);

The following example sorts the elements of the $a array in ascending alphabetical order:

Example

$a=array(25,14,18,3,19,17,13,10);

rsort($a)

```
<?php
$a=array(25,14,18,3,19,17,13,10);
$length=count($a);
echo "Before sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
```

Before sorting
25
14
18
3
19
17
13
10
After sorting
25
19
18
17
14
13

```
 echo $a[$i]."<br/>";
}
rsort($a);
echo "After sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
 echo $a[$i]."<br/>";
}
?>
```

Functions

User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

A function will be executed by a call to the function

1. Functions without parameters (Create a User Defined Function in PHP)

A user defined function declaration starts with the word "function":

Syntax

function *functionName*()

{
  code to be executed;
}

Example

```
<?php
function writeMsg() {
   echo "Hello world!";
}

writeMsg(); // call the function
?>
```

2. Function with parameters

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the muni() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

```
<?php
function muni($fname)

 {
    echo "$fname <br>";
}
muni("Jani");
muni("Hege");
muni("Stale");
muni("Kai Jim");
muni("Borge");
?>
```

3. Returning values

To let a function return a value, use the return statement:

Example

```
<?php
function sum($x, $y)

{
   $z = $x + $y;
   return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

GET & POST Methods

There are two ways the browser client can send information to the web server.

GET Method

POST Method

1. GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

http://www.test.com/index.htm?name1=value1&name2=value2

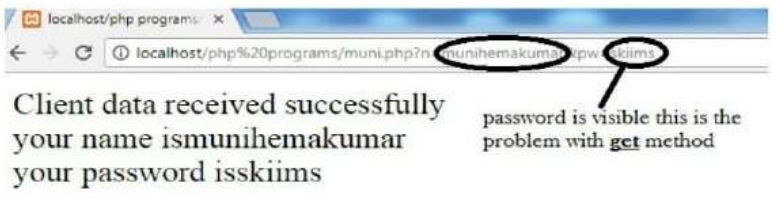
- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

getrequest.html

```
<form action="muni.php" method="get">
 Enter name<input type="text" name="n" /> <br/>
 Enter password<input type="password" name="pw" /> <br/>
 <input type="submit" value="submit" />
</form>
```

muni.php

```
<?php
 $yn=$_GET['n'];
 $ypw=$_GET['pw'];
 echo "Client data received successfully"."<br/>";
 echo "your name is".$yn."<br/>";
 echo "your password is".$ypw;
?>
```



The POST Method

- The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

postrequest.html

```
<form action="muni.php" method="post">
 Enter name<input type="text" name="n" /> <br/>
 Enter password<input type="password" name="pw" /> <br/>
 <input type="submit" value="submit" />
</form>
```

muni.php

```
<?php
 $yn=$_POST['n'];
 $ypw=$_POST['pw'];
 echo "Client data received successfully"."<br/>";
 echo "your name is".$yn."<br/>";
 echo "your password is".$ypw;
?>
```

MySql:

Introduction:

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server, developed by T.c.X. DataKonsultAB. MySQL, available for download at www.mysql.com, offers the database programmer with an array of options and capabilities rarely seen in other database servers. MySQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section, as there is charge for licensing the product.

These capabilities range across a number of topics, including the following:

a) Ability to handle an unlimited number of simultaneous users.

b) Capacity to handle 50,000,000+ records.

c) Very fast command execution, perhaps the fastest to be found on the market.

d)Easy and efficient user privilege system.

However, perhaps the most interesting characteristic of all is the fact that it's free. That's right, T.c.X offers MySQL as a free product to the general public.

Reasons to Use MySQL

a) Scalability and Flexibility

The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported.

b) High Performance

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results.

C) High Availability

Rock-solid reliability and constant availability are hallmarks of MySQL, with customers relying on MySQL to guarantee around-the-clock uptime. MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

d) Robust Transactional Support

MySQL offers one of the most powerful transactional database engines on the market. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa.

e) Web and Data Warehouse Strengths

MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data inserts capability, and strong support for specialized web functions like fast full text searches.

f) Strong Data Protection

Because guarding the data assets of corporations is the number one job of database professionals, MySQL offers exceptional security features that ensure absolute data

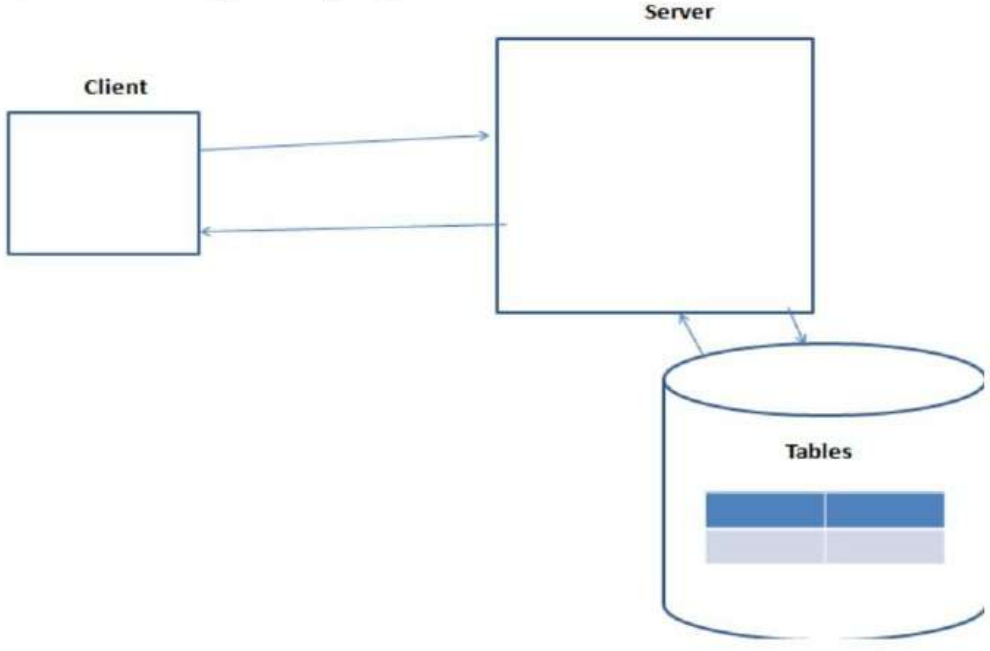
protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server, with the ability to block users down to the client machine level being possible.

g) Management Ease

MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX.

PHP Main Features of MySQL

- Tested with a broad range of different compilers.
- Works on many different platforms.
- The MySQL Server design is multi-layered with independent modules.
- Fully multi-threaded using kernel threads. It can easily use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines.
- Uses very fast B-tree disk tables with index compression.
- Relatively easy to add other storage engines. This is useful if you want to provide an SQL interface for an in-house database.
- A very fast thread-based memory allocation system.
- Very fast joins using an optimized one-sweep multi-join.
- In-memory hash tables, which are used as temporary tables.
- SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.
- The server is available as a separate program for use in a client/server networked environment.



Database Connectivity (PHP Database Interaction in FIVE steps)

PHP has a pretty straight forward method to working with MySQL databases. I will explain to establish a connection to the MySQL database, but I am not going to go into SQL which is the language used to put data in and get data out of a database. There are five steps to make PHP database interaction and I will explain each one in detail with example code:

1. Create connection
2. Select database
3. Perform database query
4. Use return data
5. Close connection

1. Create connection

It is very important when you want to start a dynamic website to create connection with your SQL (we are talking about mysql) by using mysql_connect() function. In mysql_connect() arguments should be string and it's important to use die() function with mysql_error() function to check if there is a problem with the connection or not.

2. Select database

Now we have to select the database which we are creating and saving our data by using mysql_select_db() function which the arguments are the name of database and connection we made earlier.

3. Perform database query

In this step we have to select out data from database and bring it into our web page. The best decision to use mysql_query() function which it will Send a MySQL query with 2 arguments as displayed in the above code.

4. Use return data

To display the result on your web page, you have to use while() loop function in addition to mysql_fetch_array() function which fetch a result row as an associative array, a numeric array, or both.

5. Close connection

To close connection, it is better to use mysql_close() function to close MySQL connection. This is a very important function as it closes the connection to the database server. Your script will still run if you do not include this function. And too

many open MySQL connections can cause problems for your account. This it is a good practice to close the MySQL connection once all the queries are executed.

Example

// Five steps to PHP database connections:

// 1. Create a database connection

```
// (Use your own servername, username and password if they are different.)
// $connection allows us to keep referring to this connection after it is established
$connection = mysql_connect("localhost","root","myPassword");
if (!$connection)
{
die("Database connection failed: " . mysql_error());
}
```

// 2. Select a database to use

```
$db_select = mysql_select_db("widget_corp",$connection);
if (!$db_select)
{
die("Database selection failed: " . mysql_error());
}
```

// 3. Perform database query

```
$result = mysql_query("SELECT * FROM subjects", $connection);
if (!$result)
{
die("Database query failed: " . mysql_error());
}
```

// 4. Use returned data

```
while ($row = mysql_fetch_array($result))
{
echo $row["menu_name"]." ".$row["position"]."<br />";
}
```

// 5. Close connection

```
mysql_close($connection);
```

MySql Methods

1. mysqli_num_rows()

Definition and Usage