Unit: 6
PHP

Outline

Introduction to PHP

- PHP is a scripting language that allows you to create dynamic Web pages
- You can embed PHP scripting within normal html coding
- PHP was designed primarily for the Web
- PHP includes a comprehensive set of database access functions
- High performance/ease of learning/low cost
- Open-source
  - Anyone may view, modify and redistribute source code
  - Supported freely by community
- Platform independent

Basics of PHP

- PHP files end with .php, you may see .php3 .phtml .php4 as well
- PHP code is contained within tags

  <?php ?> **or**

  Short-open: <? ?>
- HTML script tags: (This syntax is removed after PHP 7.0.0)

  <script language="php"> </script>
- Comments

  // for single line

  /* */ for multiline

PHP Basic Example

```
<?php
   $name = "Arjun Bala";    // declaration
?>

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Untitled document</title>
   </head>

   <body style = "font-size: 2em">
      <p>
         <strong>

            <!-- print variable name’s value -->
            Welcome to PHP, <?php print( "$name" ); ?>!
         </strong>
      </p>
   </body>
</html>
```

Scripting delimiters

Declare variable `$name`

Single-line comment

Function `print` outputs the value of variable `$name`

Variables

- All variables begin with **$** and can contain letters, digits and underscore (variable name can not begin with digit)
- PHP variables are Case-sensitive
- Don’t need to declare variables
- The value of a variable is the value of its most recent assignment
- Variables have no specific type other than the type of their current value
- Can have variable variables $$variable **(not recommended)**

Example :

```
$a = “b”;
$b = “XYZ”;
echo($$a);
```

Variables cont..

- Variable names inside strings replaced by their value

  Example : $name = “XYZ”;

  $str = “My name is $name”;

- Type conversions
  - settype function
  - Type casting
- Concatenation operator
  - . (period)

Variable types

Data Type	Description
int, integer	Whole numbers (i.e., numbers without a decimal point).
float, double	Real numbers (i.e., numbers containing a decimal point).
string	Text enclosed in either single ('') or double ("") quotes.
bool, boolean	True or false.
array	Group of elements.
object	Group of associated data and methods.
resource	An external data source.
NULL	No value.

Variables Scope

- Scope refers to where within a script or program a variable has meaning or a value
- Mostly script variables are available to you anywhere within your script.
- Note that variables inside functions are local to that function and a function cannot access script variables which are outside the function even if they are in the same file.
- The modifiers global and static allow function variables to be accessed outside the function or to hold their value between function calls respectively

Variables (Example)

```
<!-- Demonstration of PHP data types -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>PHP data types</title>
   </head>

   <body>

      <?php

         // declare a string, double and integer
         $testString = "3.5 seconds";
         $testDouble = 79.2;
         $testInteger = 12;
      ?>

```

Assign a string to variable `$testString`

Assign a double to variable

Assign an integer to variable `$testInteger`

Example (Variables) cont..

```
      <!-- print each variable’s value -->
      <?php print( $testString ); ?> is a string.<br />
      <?php print( $testDouble ); ?> is a double.<br />
      <?php print( $testInteger ); ?> is an integer.<br />

      <br />
      Now, converting to other types:<br />
      <?php

         // call function settype to convert variable
         // testString to different data types
         print( "$testString" );
         settype( $testString, "double" );
         print( " as a double is $testString <br />" );
         print( "$testString" );
         settype( $testString, "integer" );
         print( " as an integer is $testString <br />" );
         settype( $testString, "string" );
         print( "Converting back t
            $testString <br
```

Print each variable’s value

Call function `settype` to

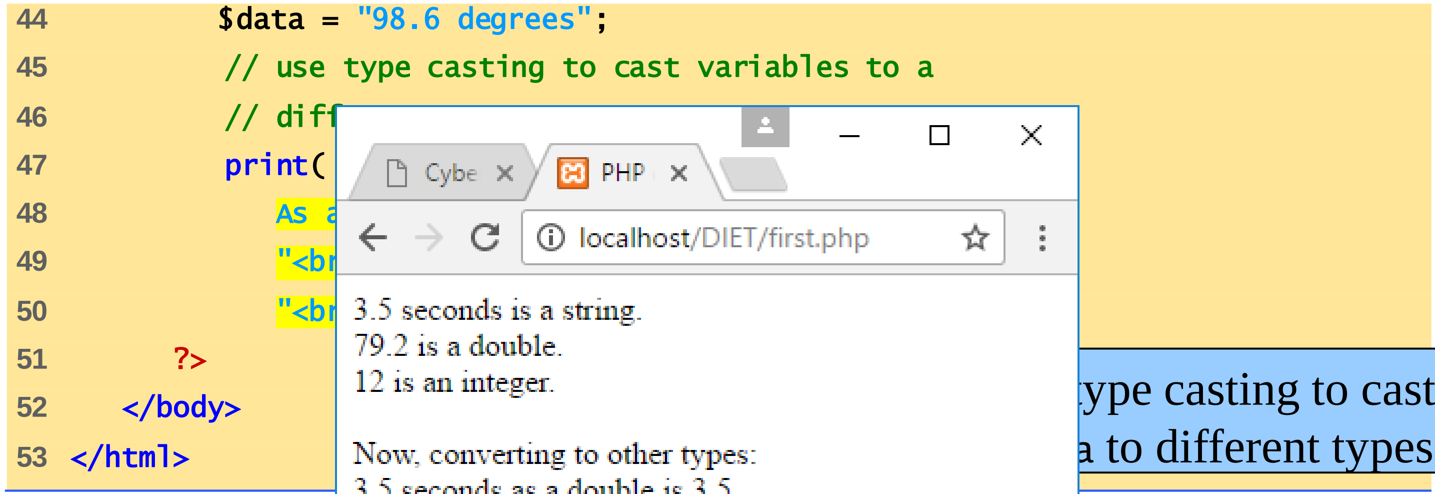
f

) a

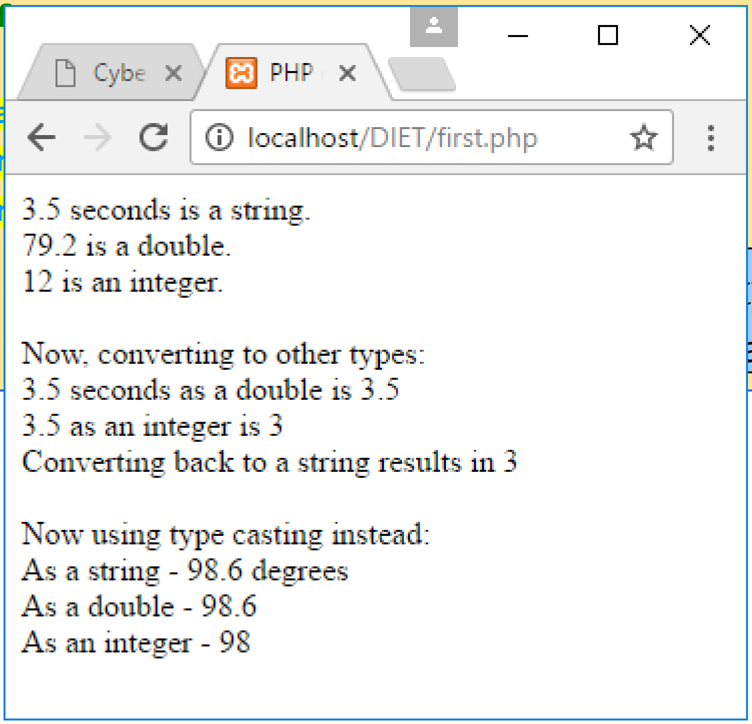
Call function `settype` to convert the data type of

Convert variable `$testString` back to a string

Example (Variables) cont..



```
            $data = "98.6 degrees";
            // use type casting to cast variables to a
            // diff
            print(
               As a
               "<br
               "<br
         ?>
      </body>
   </html>
```



```
3.5 seconds is a string.
79.2 is a double.
12 is an integer.

Now, converting to other types:
3.5 seconds as a double is 3.5
3.5 as an integer is 3
Converting back to a string results in 3

Now using type casting instead:
As a string - 98.6 degrees
As a double - 98.6
As an integer - 98
```

ype casting to cast variable
a to different types

PHP Arrays

- Array is a group of variable that can store multiple values under a single name.
- In PHP, there are three types of array.
  - Numeric Array

    These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.
  - Associative Array

    The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.
  - Multidimensional Array

    A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

PHP Arrays (Example)

```
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Array manipulation</title>
   </head>

   <body>
      <?php

         // create array first
         print( "<strong>Creating the first array</strong>
            <br />" );
         $first[ 0 ] = "zero";
         $first[ 1 ] =
         $first[ 2 ] =
         $first[] = "t

         // print each element's index and value
         for ( $i = 0; $i < count( $first ); $i++ )
            print( "Element $i is $first[$i] <br />" );
```

Create the array `$first` by assigning a value to an array element.

Assign a value to the array, omitting the index. A of the ar

Use a `for` loop to print out each element's index and value. Function `count` returns the total number of elements in the array.

PHP Arrays (Example) cont..

```
        print( "<br /><strong>Creating th
           </strong><br />" );

        // call function array to create
        $second = array( "zero", "one", "two", "three" );
        for ( $i = 0; $i < count( $second ); $i++ )
           print( "Element $i is $second[$i] <br />" );

        print( "<br /><strong>Creating the third array
           </strong><br />" );

        // assign values to non-numerical indices
        $third[ "ArtTic" ] = 2
        $third[ "LunaTic" ] =
        $third[ "GalAnt" ] = 23;

        // iterate through the ar
        // element's name and value
        for ( reset( $third ); $element = key( $third );
           next( $third ) )
           print( "$eleme
```

Call function `array` to create an array that contains the arguments passed to it. Store the array in variable `$second`.

Assign values to non-numerical indices in array `$third`.

Function `reset` sets the internal pointer to the first element of the array.

Function key returns the index of the element

Function `next` moves the internal pointer to the next element.

PHP Arrays (Example) cont..

```
         print( "<br /><strong>Creating the fourth array
            </strong><br />" );

         // call function array to create array fourth using
         // string indices
         $fourth = array(
            "January"   => "first",   "February" => "second",
            "March"     => "third",
            "May"       => "fifth",
            "July"      => "seventh
            "September" => "ninth",
            "November"  => "elevent
            );

         // print each element’s name and value
         foreach ( $fourth as $element => $value )
            print( "$element is the $value month <br />" );
      ?>
   </body>
</html>
```

Operator **=>** is used in function `array` to assign each element a string index. The value to the left of the operator is the array index, and the value to the right is the element’s value.

PHP Array Functions

- count($array) / sizeof($array)

  Count all elements in an array, or something in an object

- array_shift($array)

  Remove an item from the start of an array and shift all the numeric indexes

- array_pop($array)

  Remove an item from the end of an array and return the value of that element

- array_unshift($array,”New Item”)

  Adds item at the beginning of an array

- array_push($array,”New Item”)

  Adds item at the end of an array

PHP Array Functions cont..

- sort($array [, $sort_flags])
  - Array can be sorted using this command, which will order them from the lowest to highest
  - If there is a set of string stored in the array they will be sorted alphabetically.
  - The type of sort applied can be chosen with the second optional parameter $sort_flags which can be

• SORT_REGULAR	compare items normally (don't change type)
• SORT_NUMERIC	compare items numerically
• SORT_STRING	compare items as string
• SORT_LOCALE_STRING	compare items as string, based on the current locale

- rsort($array [, $sort_flags])
  - It will sort array in reverse order (i.e. from highest to lowest)

PHP Array Functions cont..

- shuffle($array)

  It will mix items in an array randomly.

- array_merge($array1,$array2)

  It will merge two arrays.

- array_slice($array,$offset,$length)

  returns the sequence of elements from the array $array as specified by the $offset and $length parameters.

PHP Functions

- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- Creating PHP function

  Begins with keyword function and then the space and then the name of the function then parentheses”()” and then code block “{}”

```
<?php
        function functionName()
        {
                //code to be executed;
        }
?>
```

Note: function name can start with a letter or underscore "_", but not a number!

PHP Functions cont..

- Where to put the function implementation?

  In PHP a function could be defined before or after it is called.

```
<?php
	function functionName()
	{
		//code to be executed;
	}

	functionName();
?>
```

Here function call is after implementation, which is valid

```
<?php
	functionName();

	function functionName()
	{
?>
```

Here function call is before implementation, which is also valid

Browser Control

- PHP can control various features of a browser.
- This is important as often there is a need to reload the same page or redirecting the user to another page.
- Some of these features are accessed by controlling the information sent out in the HTTP header to the browser, this uses the header() command such as :

```
header(“Location: index.php”);
```

- We can also control the caching using same header() command

```
header(“Cache-Control: no-cache”);
```

Or can specify the content type like,

```
header(“Content-Type: application/pdf”);
```

Browser Detection

- The range of devices with browsers is increasing so it is becoming more important to know which browser and other details you are dealing with.
- The browser that the server is dealing can be identified using:

  ```
  $browser_ID = $_SERVER['HTTP_USER_AGENT'];
  ```

- Typical response of the above code is follows:

  Mozilla/5.0 (**Windows NT 10.0**; Win64; **x64**) AppleWebKit/537.36 (KHTML, like Gecko) **Chrome**/56.0.2924.87

  - which specifies that user is using **Chrome** browser and **windows 10** OS with **64 bit** architecture

PHP String Functions

- Most of the time in PHP we suppose to do manipulation of strings, wheatear it be input from the user, databases or files that have been written.
- String can be think as a array of characters, so it is possible to do something like this,

  $mystring = “Welcome to Vidush Somany Institute of Technology and Research”;

  print ($mystring[11]) ; // which will print ‘**V**’

  - This uses an index as an offset from the beginning of the string starting at **0**
- Often, there are specific things that need to be done to a string, such as reversing, extracting part of it, finding a match to part or changing case etc..

PHP String Functions cont..

String Function	Purpose
strlen($string)	Returns length of string.
strstr($str1,$str2)	Finds str2 inside str1 (returns false if not found or returns portion of string1 that contains it)
strpos($str1,$str2)	Finds str2 inside str1 and returns index.
str_replace($search,$replace,$str[,$count])	Looks for $search within $str and replaces with #replace, returning the number of times this is done in $count
substr($string,$startposition[,$endposition])	Returns string from either start position to end or the section given by $startpos to $endpos

trim($string)
rtrim($string)
ltrim($string)
	Trims away white space, including tabs, newlines and spaces, from both beginning and end of a string. **ltrim** is for the start of a string only and **rtrim** for the end of a string only

PHP String Functions cont..

String Function	Purpose
strip_tags($string,$tags)	Strips out HTML tags within a string, leaving only those within $tags intact
stripslashes($string)	Strips out inserted backslashes
explode($delimiters,$string)	It will breaks $string up into an array at the points marked by the $delimiters
implode($array)	Function returns combined string from an array.
strtolower($string)	Converts all characters in $string to lowercase.
strtoupper($string)	Converts all characters in $string to uppercase.
ucword($string)	Converts all the first letters in a string to uppercase.

Form Processing

- We can access form data using there inbuilt PHP associative array.
  - $_GET => in case we have used **get** method in the form
  - $_POST => in case we have used **post** method in the form
  - $_REQUEST => in both the cases

- For example,

html

```
<form action="recive.php"
method="get">
    <input type="text"
name="UserName">
    <input type="submit">
</form>
```

recive.php

```
<?php
   $u = $_GET['UserName'];
   echo($u);
?>
```

File Handling in PHP

- PHP has several functions for creating, reading, uploading, and editing files.
- fopen($filename, $mode) will return the handle to access file.
    - "r" (Read only. Starts at the beginning of the file)
    - "r+" (Read/Write. Starts at the beginning of the file)
    - "w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist)
    - "w+" (Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist)
    - "a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist)
    - "a+" (Read/Write. Preserves file content by writing to the end of the file)

File Handling in PHP cont..

Function	Purpose
file_exists($file)	Will return true if file is found, false otherwise
filesize($file)	Returns the size of the file in bytes.
fread($file,$bytesToRead)	Will read $bytesToRead from $file handle
fwrite($file,$str)	Will write $str in the $file handle
fclose($file)	Will close the $file handle
copy($source,$destination)	Will copy from $source to $destination
rename($oldname,$newname)	Will rename the file to $newname
unlink($file)	Will delete the file

File Handling Example

text.txt

Hello World From VSITR College

Read File

```
<?php
	$file = fopen("text.txt","a+");
	$text = fread($file,filesize("text.txt"));
	echo($text);
?>
```

Write File

```
<?php
	fwrite($file," New Content");
	$text = fread($file,filesize("text.txt"));
	echo($text);
?>
```

Cookies in PHP

- HTTP cookies are data which a server-side script sends to a web client to keep for a period of time.
- On every subsequent HTTP request, the web client automatically sends the cookies back to server (unless the cookie support is turned off).
- The cookies are embedded in the HTTP header (and therefore not visible to the users).
- **Shortcomings/disadvantages of using cookies to keep data**
  - User may turn off cookies support.
  - Users using the same browser share the cookies.
  - Limited number of cookies (20) per server/domain and limited size (4k bytes) per cookie
  - Client can temper with cookies

Cookies in PHP cont..

- To set a cookie, call setcookie()
  - e.g., **setcookie('username', ‘AVB');**
- To delete a cookie (use setcookie() without a value)
  - e.g., **setcookie('username');**
- To retrieve a cookie, refer to $COOKIE
  - e.g. **$username = $_COOKIE['username‘];**
- Note :
  - Cookies can only be set before any output is sent.
  - You cannot set and access a cookie in the same page. Cookies set in a page are available only in the future requests.

Cookies in PHP cont..

setcookie(name, value, expiration, path, domain, secure, httponly)

- Expiration
  - Cookie expiration time in seconds
  - 0 ➔ The cookie is not to be stored persistently and will be deleted when the web client closes.
  - Negative value ➔ Request the web client to delete the cookie
  - e.g.: setcookie('username', 'Joe', time() + 1800);   // Expire in 30 minutes
- Path
  - Sets the path to which the cookie applies. (Default is '/')
- Domain
  - The domain that the cookie is available.
- Secure
  - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Session in PHP

- Session is a way to make data accessible across the various pages of an entire website is to use a PHP Session.
- A session creates a file in a temporary directory on the server where registered session variables and their values are stored.
- The **location** of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**.
- When a session is started following things happen
  - PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
  - A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
  - A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_, sess_3c7foj34c3jj973hjkop2fc937e3443.

Starting a PHP Session

- A PHP session is easily started by making a call to the session_start() function.This function first checks if a session is

```
<?php
            session_start();
            if( isset( $_SESSION['counter'] ) ) {
                        $_SESSION['counter'] += 1;
            }else {
                        $_SESSION['counter'] = 1;
            }
            $msg = "You have visited this page ".  $_SESSION['counter'];
            $msg .= "in this session.";
?>
      <html><head>
          <title>Setting up a PHP session</title>
        </head><body>
          <?php  echo ( $msg ); ?>
        </body></html>
```

Destroying a PHP Session

- A PHP session can be destroyed by session_destroy() function.
- This function does not need any argument and a single call can destroy all the session variables.

Logout.php

```
<?php
            session_destroy();
?>
```

- If you want to destroy a single session variable then you can use unset() function to unset a session variable.

Logout.php

```
<?php
            unset(S_SESSION[‘counter’]);
?>
```

Unit: 6
PHP

Outline

Connection to the Server

- To connect with PHP the mysql_connect command can be used

```
<?php
	mysql_connect(server,username,password);
?>
```

- You may use the code fragment :

```
<?php
	$connect = mysql_connect(“localhost”,“root”,“”);
?>
```

Creating a Database

- To create a database, use the SQL command :

```
SQL Query
CREATE DATABASE databasename
```

- You may use the code fragment :

```
<?php
	mysql_connect(“localhost”,“root”,“”);
	$sql = “CREATE DATABASE DemoDatabase”;
	mysql_query($sql);
?>
```

Selecting a Database

- Before actually use a database we need to select it as below

```
<?php
        mysql_select_db("DemoDB") or die ("Can not Select Database");
?>
```

- The **die** command stops further script processing with an error message if the database cannot be selected.

Listing Database

- In SQL show databases will display all the current databases.

```
SQL Query
SHOW DATABASES
```

- One way to do the same in PHP is :

```
<?php
	$result = mysql_list_dbs($connect);
	for($row=0;$row<mysql_num_rows($result);$row++)
	{
		$dbases .= mysql_tablename($result,$row) . “<br/>”;
	}
	echo($dbases);
?>
```

- Note: mysql_tablename is generalized function for table/dastabase name
- Note: mysql_tablename is deprecated function and should not be used

Listing Table Names

- In SQL show tables will display all the current tables.

SQL Query

```
SHOW TABLES
```

- One way to do the same in PHP is :

```
<?php
        $result = mysql_list_tables($db);
        for($row=0;$row<mysql_num_rows($result);$row++)
        {
                $tables .= mysql_tablename($result,$row) . “<br/>”;
        }
        echo($tables);
?>
```

- Note: mysql_tablename is generalized function for table/dastbase name
- Note: mysql_tablename is deprecated function and should not be used

Creating a Database Table

- A database table is made in both the PHP and directly with the same monitor using basically the same command, which is

```
SQL Query
CREATE TABLE tablename (fileds);
```

- In case of PHP the command is actually made up as a string and then submitted using the mysql_query command :

```
<?php
      $sql =             "CREATE TABLE demo (
                         id INT NOT NULL AUTO_INCREMENT,
                         name VARCHAR(50) NOT NULL,
                         PRIMARY KEY(id))";
      mysql_query($sql);
?>
```

Inserting Data

- Data is inserted into a database table in the MySQL monitor using insert query:

```
SQL Query
INSERT INTO demo (id,name) values (NULL,'avb')
```

- In case of PHP the command is actually made up as a string and then submitted using the mysql_query command :

```
<?php
        $sql =  "INSERT INTO demo (id,name) values (NULL,'$name')";
        mysql_query($sql);
?>
```

Altering Tables

- To alter table using database monitor we can use following SQL query :

SQL Query

```
ALTER TABLE demo modify name varchar(30);
```

- In case of PHP the command is actually made up as a string and then submitted using the mysql_query command :

```
<?php
        $sql =  "ALTER TABLE demo modify name varchar(30)";
        mysql_query($sql);
?>
```

Deleting Databases

- To delete the database we can use following SQL Query

```
SQL Query
DROP DATABASE dbname
```

- In case of PHP the command is actually made up as a string and then submitted using the mysql_query command :

```
<?php
      $sql =  “DROP DATABASE dbname”;
      mysql_query($sql);
?>
```

Select Queries

- Once we have set up a database with information, queries can be applied to actually find information

SQL Query

SELECT * FROM demo where ……………………

- In case of PHP the command is actually made up as a string and then submitted using the mysql_query command, it will return an result which can then be manipulated to get desired output:

```
<?php
        $sql =  “select * from demo”;
        $result = mysql_query($sql);
?>
```

Accessing the Result

- We have 3 methods to access the result generated from the select query :
  - $row = mysql_fetch_row($result)

    Result can be accessed using **numeric** index starting from **0.**

    Ex : echo($row[**1**]); // in our demo table it will return name as it is in 1 index.
  - $row = mysql_fetch_assoc($result)

    Result can be accessed using **string** index where index is the column name.

    Ex : echo($row['**name**']);
  - $row = mysql_fetch_array($result)

    Result can be accessed using **numeric** as well as **string** index

    Ex : echo($row[1]); // in our demo table it will return name as it is in 1 index

    Or

    Ex: echo($row['**name**']); // both will be valid and will return same

Accessing the Result cont..

```
<?php
        $sql =  “select * from demo”;
        $result = mysql_query($sql);

        while($row = mysql_fetch_row($result))
        {
                echo($row[1] . “<br/>”);
        }
                                        OR
        while($row = mysql_fetch_assoc($result))
        {
                echo($row[‘name’] . “<br/>”);
        }
                                        OR
        while($row = mysql_fetch_array($result))
        {
                echo($row[0] . “       ”. $row[‘name’] . “<br/>”);
        }
?>
```