National University of computer and emerging sciences

LAB # 11: PHP PROGRAMMING AND CRUD OPERATIONS

Objective:

I. **Web Development Basics**
II. **PHP basics**

Scope:

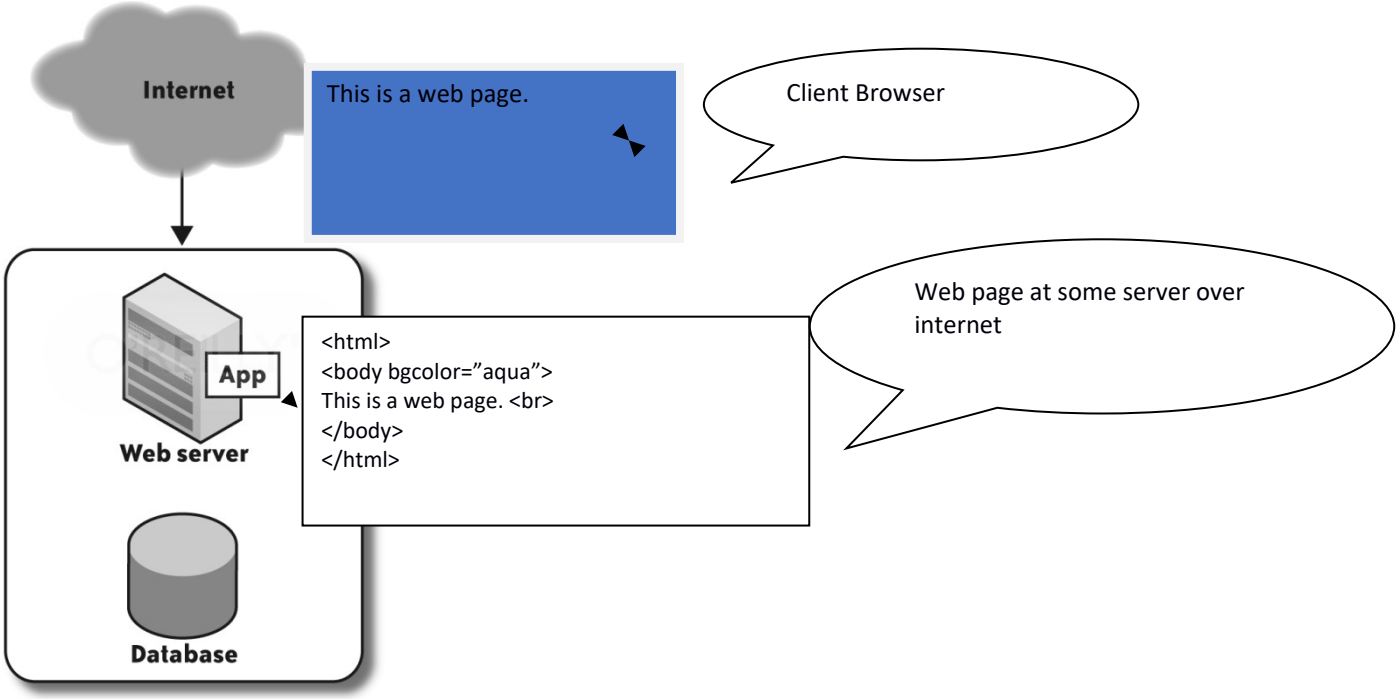
The student should know the following:

Basic Web development life cycle
Programming in PHP

1) Web Development Architecture

It typically comprises of a browser, HTML, any scripting language, web server and database server. In web development we use all of these to display some information (or build a page) to our target user over the internet. The general pattern of displaying information is as follows;
When a request for a page comes from the browser, the web server performs following steps:

- Read the request from the browser.
- Find the page on the server.
- Send the page back across the internet to the client browser.
- Client browser interprets HTML and create (display) the (information requested) as a web page.



1.1 Browser

The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

1.2 Web Server

The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets etc.

2.1 Scripting Language PHP

- PHP stands for PHP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

PHP works in a similar way to JSP and ASP: scripts sections are enclosed in <?php .. ?> tags and embedded within an HTML page. These scripts are executed on the server before the page is sent to the browser, so there is no issue of browser support for PHP page.

2.2 Benefits of Server Side processing

1. Minimizes network traffic.
2. Make quicker downloading since the client browser only downloading HTML page.
3. Provides improved security measure, since we can code things that can never be viewed from the browser.

2.3 What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php" or ".php3"

2.4 Where to Start?

To get access to a web server with PHP support, you can:

1) Install Apache server on your own machine, install PHP and ready to go.

3. PHP Block

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.
On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.
For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

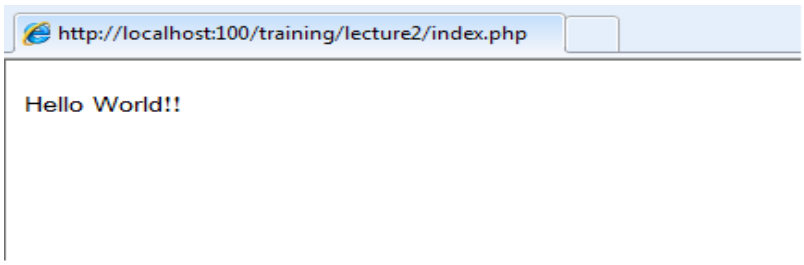
```
<?php

?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code. Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.
There are two basic statements to output text with PHP: **echo** and **print**. In the following example we'll be using **echo** statement to output the text "Hello World".
Create a web page (blank page) with index.php in your htdocs/training/ directory.
Now Edit the file and Put the following code.

Example 3.1

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

Note: The file must have a .php extension. If the file has .html extension, the PHP code will not be executed.
Run the page in the browser by typing http://localhost/training/index.php



3.1 Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>
```

```
<?php
  //This is a comment
  /*
   This is
   a comment
   block
  */
?>
</body>
</html>
```

4. Declaring Variables in PHP

Variables are used for storing values, like text strings, numbers or arrays.
When a variable is declared, it can be used over and over again in your script.
All variables in PHP start with a $ sign symbol.
The correct way of declaring a variable in PHP:
$var_name = value;
Let's try creating a variable containing a string, and a variable containing a number:

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

In PHP, a variable does not need to be declared before adding a value to it.
In the example above, you see that you do not have to tell PHP which data type the variable is.
PHP automatically converts the variable to the correct data type, depending on its value.

4.1 Naming Rules for Variables

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string), or with capitalization ($myString)

Now open your index.php and add following (bold) text to it.

Example 4.1

```
<html>
<body>
<?php
   echo "Hello World";
   echo “<br>”;
```

```
  $txt="Hello World";
  echo $txt;
?>
</body>
</html>
```

The output of the code above will be:

```
Hello World
Hello World
```

Note the difference (can anybody explain it for others).

4.2 The Concatenation Operator

There is only one string operator in PHP.
The concatenation operator (.) is used to put two string values together.
To concatenate two string variables together, use the concatenation operator:

```
<?php
  $txt1="Hello World!";
  $txt2="What a nice day!";
  echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

```
Hello World! What a nice day!
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

5. HTML Form handling using PHP

When dealing with HTML forms it is to be noted that all forms element in an HTML page will **automatically** be available to your PHP scripts.

Example 5.1

Now create another file user.php and add following (bold) text to it.
This will contains an HTML form with two input fields and a submit button

```
<html>
<body>
<form action="user_reply.php" method="post">
Name: <input type="text" name="fname" />
```

```
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

When a user fills out the form above and click on the submit button, the form data is sent to a PHP file, called "user_reply.php".

5.1 The $_GET

The built-in $_GET is used to collect values from a form sent with method="get".

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.

```
http://localhost:100/training//user_reply.php?
name=test&gender=Male&education=Masters&submit=Submit
```

5.1.1 When to use method="get"?

When using method="get" in HTML forms, all variable names and values are displayed in the URL.
Note: This method should not be used when sending passwords or other sensitive information!
However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
Note: The get method is not suitable for very large variable values. It should not be used with values exceeding 2000 characters.

5.2 The $_POST

The built-in $_POST is used to collect values from a form sent with method="post".
Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

```
http://localhost:100/training/lecture2/user_reply.php
```

Note: However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).

5.2.1 When to use method="post"?

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send. This method should be used when sending passwords or other sensitive information.
However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Example 5.2

Now create another file user_reply.php and add following (bold) text to it.

```
<html>
<body>
Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

Output could be something like this:

```
Welcome [the name which user provided]
You are [the age which user provided] years old.
```

Example 5.3

Using short output notation (change the text in user_reply.php file as follows).

```
<html>
<body>
Welcome <?=$_POST["fname"];?>!<br />
You are <?= $_POST["age"];?> years old.
</body>
</html>
```

The output will be same. Note the difference here.

5.3 The PHP $_REQUEST

The PHP built-in $_REQUEST contains the contents of both $_GET and $_POST.
The $_REQUEST function can be used to collect form data sent with both the GET and POST methods.
For example, Example 3.3 output can also be achieved as follows;

Example 5.4

```
<html>
```

```
<body>
Welcome <?=$_REQUEST["fname"];?>!<br />
You are <?= $_REQUEST["age"];?> years old.
</body>
</html>
```

6. PHP Sessions

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

6.1 PHP Session Variables

When you are working 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 there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.
A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.
Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

6.2 Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.
Note: The session_start() function must appear BEFORE the <html> tag:

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

<html>
<body>

</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

6.3 Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php
   session_start();
   // store session data
   $_SESSION['views']=1;
?>
<html>
<body>
<?php
   //retrieve session data
   echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

Output:

```
Pageviews=1
```

In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
   session_start();
   if(isset($_SESSION['views']))
   $_SESSION['views']=$_SESSION['views']+1;
   else
   $_SESSION['views']=1;
   echo "Views=". $_SESSION['views'];
?>
```

6.4 Destroying a Session

If you wish to delete some session data, you can use the unset() or the session_destroy() function.
The unset() function is used to free the specified session variable:

```
<?php
   unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

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

Note: session_destroy() will reset your session and you will lose all your stored session data.

6. PHP Operators

Operators are used to operate on values.
The following tables list the different operators used in PHP.

6.1 Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

6.2 Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y

.=	x.=y	x=x.y
%=	x%=y	x=x%y

6.3 Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

6.4 Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

7. PHP if else statement

Conditional statements are used to perform different actions based on different conditions.
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.
In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false

- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

7.1 The 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;}*
The following example will output "Have a nice weekend!" if the current day is Friday:

Example 7.1

```
<html>
<body>
<?php
   $d=date("D");
   echo $d;
   if ($d=="Fri")
   {
    echo "Have a nice weekend!!";
   }
?>
</body>
</html>
```

Notice that there is no ..else.. in this syntax. The code is executed **only if the specified condition is true**.

7.2 The if...else Statement

Use the if....else statement to execute some code if a condition is true and another code if a condition is false.
Syntax
if (*condition*)
 {
code to be executed if condition is true;

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

Example 7.2

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
   $d=date("D");
   if ($d=="Fri")
   {
     echo "Have a nice weekend!";
   }
   else
    echo "Have a nice day!";
?>
</body>
</html>
```

7.3 The if...elseif....else Statement

Use the if....elseif...else statement to select one of several blocks of code to be executed.
Syntax
if (*condition*)
 {*code to be executed if condition is true;}*
elseif (*condition*)
 {*code to be executed if condition is true;}*
else
 {*code to be executed if condition is false;* }

Example 7.3

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Monday!" if the current day is Monday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
```

```
if ($d=="Fri")
 {echo "Have a nice weekend!";}
elseif ($d=="Mon")
 {echo "Have a nice Monday!";}
else
 {echo "Have a nice day!";}
?>
</body>
</html>
```

7.4 The Switch Statement

Conditional statements are used to perform different actions based on different conditions. Use the switch statement to select one of many blocks of code to be executed.

Syntax

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

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 7.4

```
<html>
<body>
<?php
$x=3;
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
```

```
 break;
case 3:
 echo "Number 3";
 break;
default:
 echo "No number between 1 and 3";
}
?>
</body>
</html>
```

Example 7.5

Use of input box and form with switch choices

```
<html>
<body>
<form action="" method="post">
Enter Number:<input type="text" name="num" /><br>
<br>
<input type="submit" name="button" value="submit"/>
<input type="submit" name="button" value="show range" />
</form>
</body>
</html>
 <?php
if(isset($_POST["button"]))
{
  $btn = $_POST["button"];
  if($btn=="submit")
  {
     if(isset($_POST["num"]))
     {
          switch($_POST["num"])
          {
                case 1:
                      echo "<h1>Number is 1</h1>";
                      break;
                case 2:
                      echo "Number is 2";
                      break;
                case 3:
                      echo "Number is 3";
                      break;
                case 4:
                      echo "Number is 4";
                      break;
                default:
```

```
                        echo "Nothing is chosen from selection 1 to 4";
                }
            }
        }
    elseif($btn=="show range")
    {
        echo "Range is 1 to 4";
    }
}

?>
```

8. PHP Arrays

An array stores multiple values in one single variable.

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="Toyota";
$cars2="Hyundai";
$cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kinds of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

8.1 Numeric Arrays

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

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

2. In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example 8.1

In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

The code above will output:

```
Saab and Volvo are Swedish cars.
```

8.2 Associative Arrays

An associative array, each ID key is associated with a value.

With associative arrays we can use the values as keys and assign values to them.

Example 8.2

In this example we use an array to assign ages to the different persons:

```
$ages = array("A"=>32, "B"=>30, "C"=>34);
```

Example 8.3

This example is the same as example 2.2, but shows a different way of creating the array:

```
$ages['A'] = "32";
```

```
$ages['B'] = "30";
$ages['C'] = "34";
```

The ID keys can be used in a script:

```
<?php
$ages['A'] = "32";
$ages['B'] = "30";
$ages['C'] = "34";

echo "A is " . $ages['B'] . " years old.";
?>
```

The code above will output:

```
A is 32 years old.
```

8.3 Multidimensional Arrays

In a multidimensional 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.

Example 8.4

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$shops = array
 (
 "flowers"=>array( "rose", "daisy", "orchid"  ),
 "prices"=>array  (  "1.5",  "0.75",  "1.15"  )
 );
```

The array above would look like this if written to the output:

```
Array
(
   [flowers] => Array
      (
         [0] => rose
         [1] => daisy
         [2] => orchid
      )

   [prices] => Array
      (
         [0] => 1.5
         [1] => 0.75
```

```
        [2] => 1.15
    )

)
```

Example 8.5

Lets try displaying a single value from the array above:

```
echo $shops['flowers'][0] .' costs '. $shops['prices'][0];
```

The code above will output:

```
rose costs 1.5
```

9. PHP 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.

In PHP, we have the following looping statements:

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

9.1 The while Loop

The while loop executes a block of code while a condition is true.

Syntax

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

Example 9.1

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
```

```
<body>
<?php
$i=1;
while($i<=5)
 {
 echo "The number is " . $i . "<br />";
 $i++;
 }
?>
</body>
</html>
```

Output:

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

9.2 The do...while Statement

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

Syntax

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

Example 9.2

The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

```
<html>
<body>
<?php
$i=1;
do
 {
 $i++;
 echo "The number is " . $i . "<br />";
 }
while ($i<=5);
?>
```

```
</body>
</html>
```

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

9.3 The for Loop

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

Syntax

```
for (init; condition; increment)
  {
  code to be executed;
  }
```

Parameters:

- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

Note: Each of the parameters above can be empty, or have multiple expressions (separated by commas).

Example 9.3

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>
```

```
</body>
</html>
```

Output:

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

9.4 The foreach Loop

The foreach loop is used to loop through arrays.

Syntax

```
foreach ($array as $value)
  {
  code to be executed;
  }
```

For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

Example 9.4

The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>
<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>
</body>
</html>
```

Output:

```
one
two
three
```

Displaying array (from example 1.5) using for loop;

Example 9.4

```
<html>
<body>
<?php
echo "<h2>Manual access to each element from associative array</h2>";

for ($row = 0; $row < 3; $row++)
{
  echo $shops['flowers'][$row] .' costs '. $shops['prices'][$row];
   echo "<br />";
}
?>
</body>
</html>
```

PHP Functions

In this lecture we will show you how to create your own functions.

To keep the script from being executed when the page loads, you can put it into a

function. A function will be executed by a call to the function.

You may call a function from anywhere within a page.

3.1 Create a PHP Function

A function will be executed by a call to the function.

Syntax

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

PHP function guidelines:

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

Example 3.1

A simple function that writes my name when it is called:

<html>
<body>

```
<?php
function display_name()
{
echo "My name is Ali";
}
display_name ();
?>
</body>
</html>
```

Output:

```
My name is Ali
```

3.2 PHP Functions - Adding parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

Parameters are specified after the function name, inside the parentheses.

Example 3.2

The following example will write different first names:

```
<html>
<body>
<?php
function display_name($fname)
{
echo "My name is ". $fname ;
}
display_name ("Ali");
?>
</body>
</html>
```

Output:

```
My name is Ali
```

Example 3.3

The following function has two parameters:

```
<html>
<body>
<?php
function display_name($fname,$lname)
{
echo "My name is ". $fname . ' '. $lname;
}
```

```
display_name ("Ali","Ahmed");
?>
</body>
</html>
```

Output:

```
My name is Ali Ahmed
```

3.3 PHP Functions - Return values

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

```
<html>
<body>
<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}
 echo "1 + 16 = " . add(1,16);
?>
</body>
</html>
```

Output:

```
1 + 16 = 17
```

Using Databases

To use a database or query some table in a database, following are the typical steps to do;

1. Create a connection to the database.
2. If connection successful, write a query statement.
3. Parse this query using connection handler and query statement (note that this step is specific to ORACLE only).
4. Execute query.
5. Fetch the result set generated by the query.
6. Loop over the result set and display data using html and PHP.

4.1 Step 1 (Create a Connection to the database)

In PHP, this is done with the [db type]_connect()
function. [db type]_connect(servername, username, password);

Parameter	Description
servername	Specifies the server to connect to.
username	Specifies the username to log in with. Default value is the name of the user that owns the database server
password	Specifies the password to log in with.

Note: oracle 11g servername can be found in: (oraclebase)\app\Your_username\product\11.2.0\dbhome_1\NETWORK\ADMIN\ tnsnames.ora

```
ORACLR_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
    (CONNECT_DATA =
      (SID = CLRExtProc)
      (PRESENTATION = RO)
    )
  )

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = Turab-PC)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )
```

Example 4.1

Create a web page (blank page) with the name as dbcon.php in your 'htdocs' directory.

Type the following code in the file and run it in the browser.

```
<html>
<body>
<?php
// example 2.1 ..creating a database connection
$db_sid = "(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = Turab-PC)(PORT =
1521)) ) (CONNECT_DATA = (SID = orcl) ) )";

$db_user = "scott";
$db_pass = "1234";
$con = oci_connect($db_user,$db_pass,$db_sid);
if($con)
{
        echo "connection successful.";
}
else
{
         die('Could not connect: ');
}
?>
<body>
</
html>
```

In this example we store the connection in a variable ($con) for later use in the script. The "die" part will be executed if the connection fails.

So now if the connection is successful, we'll now proceed to query the EMP table as follows;

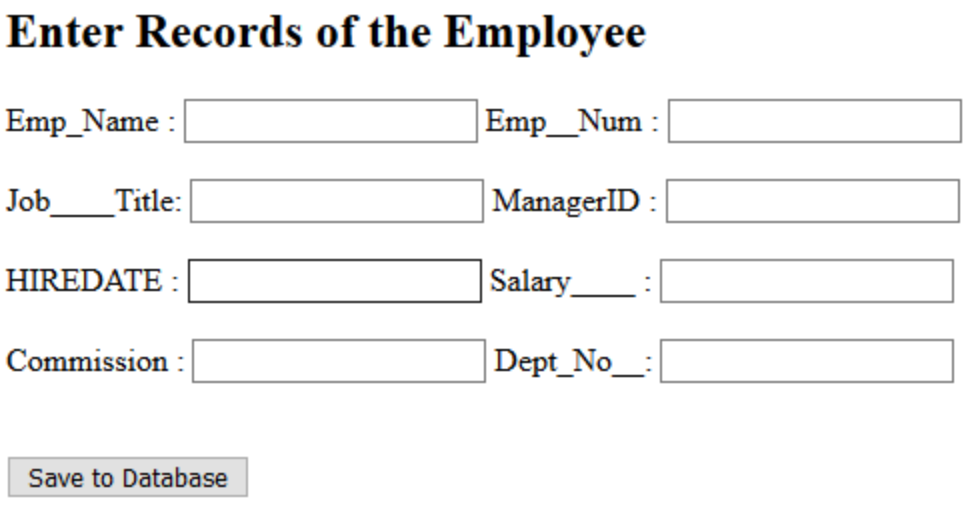
4.2 Step 2 (Write a

query) Example 4.2

Write the bold line in your file as follows;

```
<html>
<body>
<?php
// example 2.1 ..creating a database connection
  $db_sid = "(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = Turab-PC)(PORT =
1521)) ) (CONNECT_DATA = (SID = orcl) ) )";

$db_user = "scott";
$db_pass = "1234";
```

Task 1: Create PHP file (insertion.php) that takes input from user as shown in Fig. 1. User have to Enter Data in the Text fields in order to Insert Employee Name, Number, Job, Salary and so on. When user click on the "Save to Database" button, all the data should be inserted into Emp table of Scott User. And **"Insertion Successful"** Message should be displayed on the same file.



Task 2: Create PHP file (update.php) as shown in ***Fig. 2.*** that search the record on the basis of Emp_no enter by the user. When user Click on the ***Search the Record*** button, all data should be Display on the respective Fields as shown in ***Fig. 3***. User edit the Record as text fields are editable then press ***Update the Record*** Button. Record should be updated in the Database. And "Record Updated" Message should be display on the same file.

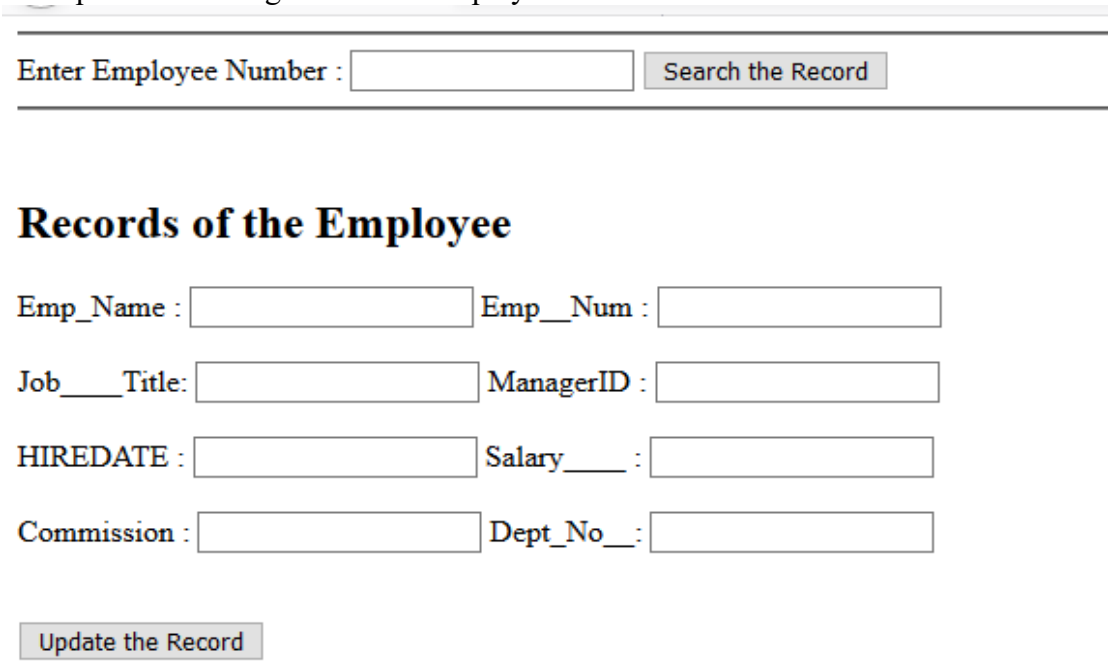


Fig. 2

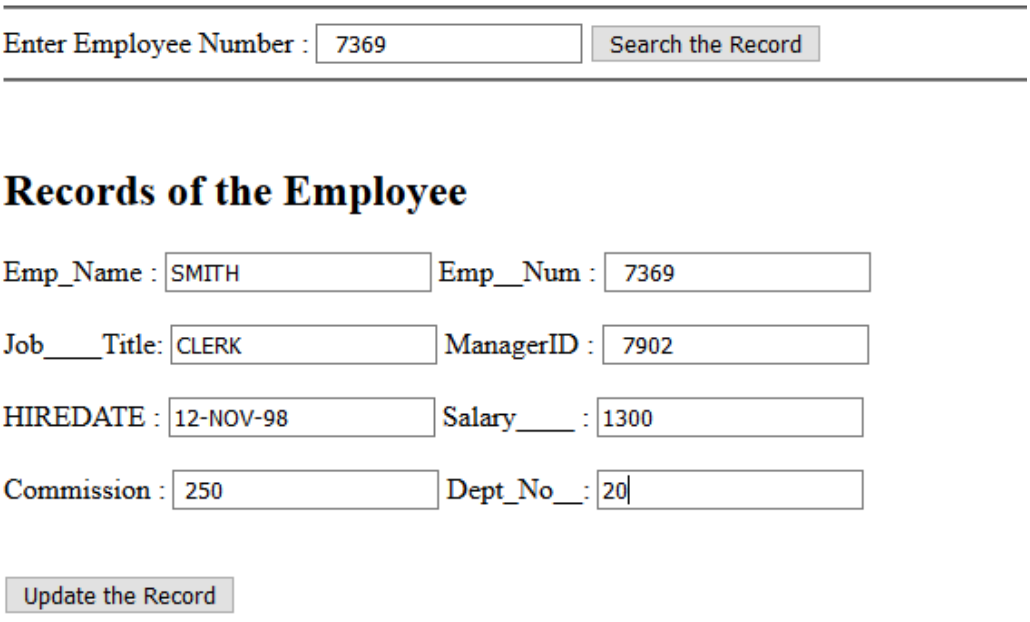


Fig. 3

TASK 3: Create PHP file (del_info.php) which receive Emp_no from user and delete the record from the database on basis of received Emp_no and output the following message “Record Deleted “on the Screen after deletion.