PHP Syllabus

- Introduction
  - Downloading, installing, configuring PHP
  - Programming in a Web environment concept
- Literals
  - Textual, numeric, boolean
- Variables and data types (casting)
  - Datatype testing, casting and conversion
- Constants
  - Define()
- Operators
  - General Operators
  - String Operators
  - Numerical Operators
  - Logical operators
- Expressions and Statements
  - if ... else ... elseif
  - switch
  - while and do ... while
  - for and foreach
  - require and include
  - exit
- Functions
  - Defining functions
  - Passing Arguments
  - Variable Scope and Lifetime
  - Using functions to organize code
- Arrays
  - Starting Simply
  - Looping Through an Array
  - A Sequentially Indexed Array
  - Non-sequentially Indexed Arrays
  - Built-in Array functions
  - Using Arrays with Form Elements
- Variables from outside world
  - System and GET variables and $HTTP_Arrays
  - POST variables
  - HTTP header variables
- Objects and OOP
  - Object-Oriented Programming
  - Defining a Class
  - Objects
  - Instantiating the Class
  - Constructors
  - Inheritance

- Handling user inputs (form)
  - Use of $_POST[ ] / $_GET[ ] to display form submitted data
  - Query String Variable and $_GET[ ]
  - Form Validation with JavaScript
  - Uploading files with $_FILES[ ]
- Sessions
  - Adding session support to PHP
  - Using PHP sessions
  - Starting sessions
  - Registering sessions variables
- File Handling
  - Opening file
  - Reading from file
  - Displaying file
  - Writing to file
  - Closing file
  - Copying deleting and renaming file
  - Built-in function on file
  - Uploading files
- Sending E-mail
  - Simple email
  - HTML formatted email
- Database (PHP and MySQL)
  - Introduction and concept
  - SQL concept
  - Database connectivity
  - MySQL built-in functions
  - Select, Insert, Update, Delete data using PHP

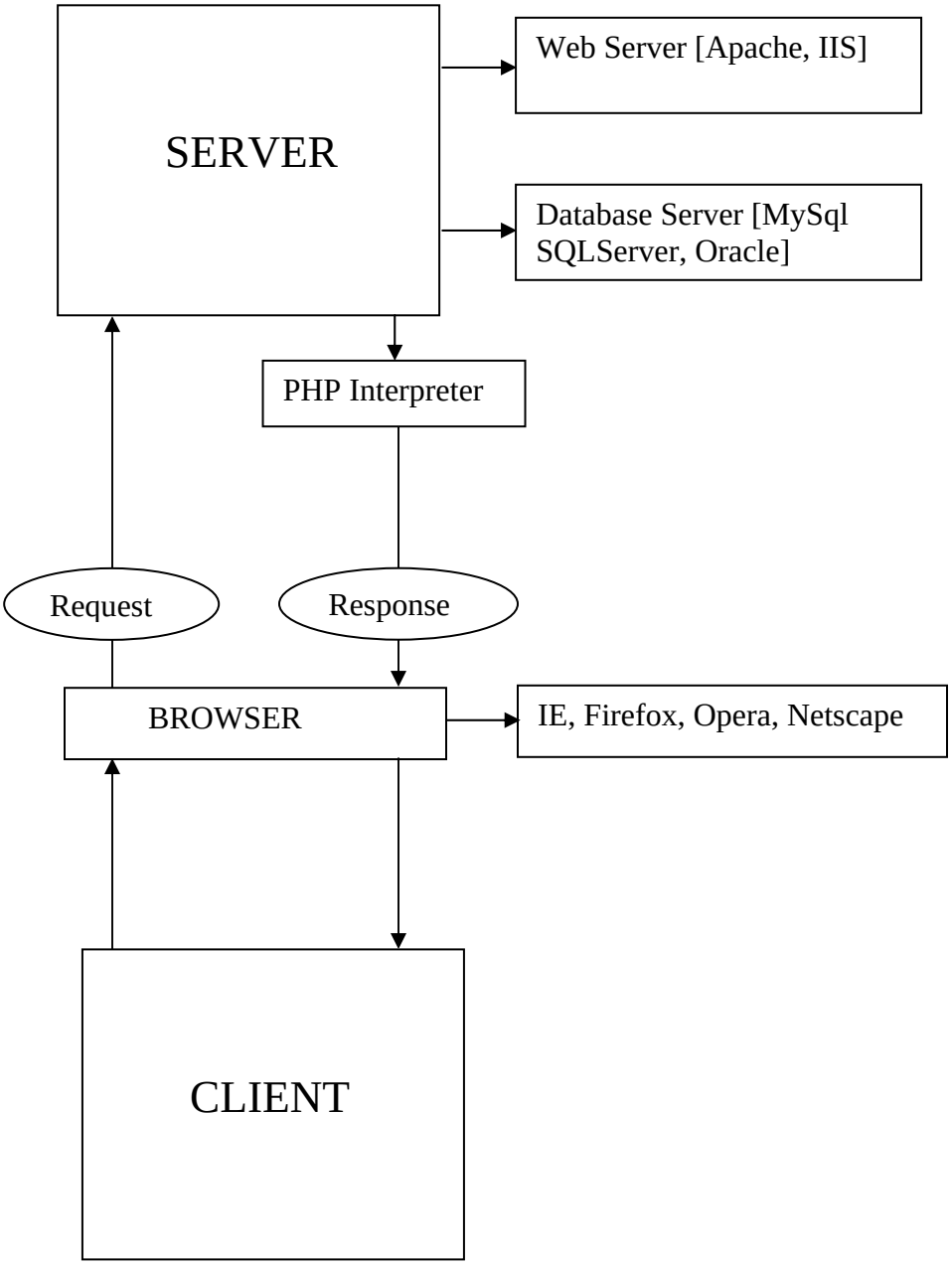

Fig: Working in web environment

PHP

- It is recursive acronyms for PHP: Hypertext preprocessor.
- It is a HTML embedded, open-source, server side scripting language.
- To design PHP pages, the recommended extensions are .php, .php3, .php4, .phtml.
- PHP enhances HTML codes using various conditional or looping structures. Whatever the code written in PHP, the final result is HTML.

 # Application of PHP:

1. Manage site content dynamically. (Both production and display)
2. To develop web delivered business application. (online payment, security, credit card processing)
3. To implement email in site
4. To develop advertising networks
5. To fill forms online and submit them
6. To conduct surveys, polls and test.
7. To develop community features like forums and bulletin boards.
8. To develop groupware and membership roles and directories.
9. To develop brochures, catalogue and information sites.
10. To implement personalization and technologies.
11. Any other web applications that needs back-end server(database server)

 # Adding PHP in HTML:

(1) Canonical Tag:

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

⟶ Used mostly when we use PHP and XML in the same page.

(2) Short-open Tag:

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

⟶ Php configuration(php.ini) should have *short_open_tag = on.*

(3) Script Tag:

```
<script language=”PHP”
      //php code here
</script>
```

(4) ASP like tag:

```
<%
    //php code here
%>
```

⟶ Php configuration(php.ini) should have *asp_tags = on.*

Variables in PHP:

Variables in php are defined using leading dollar($) sign.
Eg:

```
<?
$a=245;
$b="Hello World";
?>
```

Variables declaration in PHP is not type specific.

* Built-in Variables:

They are also called super global array.

(1) $PHP_SELF:
The filename of the currently executing script, relative to the document root.

(2) $_POST[] (or $HTTP_ POST _VARS[]):
An associative array of variables passed to the current script via the HTTP POST method.

(3) $_GET[] (or $HTTP_ GET _VARS[]):
An associative array of variables passed to the current script via the HTTP GET method.

(4) $_REQUEST[]:
An associative array merged from the GET, POST, and Cookie variables.

(5) $_SESSION[]:
An associative array of session variables passed to the current script.

(6) $_COOKIE[] (or $HTTP_COOKIE_VARS[]):
An associative array of variables passed to the current script via HTTP cookies.

(7) $GLOBALS[]:
An associative array with the name of the global variable being the key and the contents of that variable being the value of the array element.

(8) $_SERVER[] (or $HTTP_ SERVER _VARS[]):
An associative array of variables passed to the current script from the HTTP server

(9) $_FILES[] (or $HTTP_POST_ FILES []):
An associative array of variables containing information about files uploaded via the HTTP POST method.

(10) $_ENV[] (or $HTTP_ ENV _VARS[]):
An associative array of variables passed to the current script via the parent environment.

 # Displaying output in PHP:

(I) print ():
- This function is used to display the output in PHP in the browser.
- Print function supports single argument.

Eg: (a) print ("Hello World");
(b) $a=5;
print("Hello ".$a. " World");

(II) echo:
- This construct is used to display output in the browser.
- This is not exactly a function, so use of parentheses is not necessary.
- It supports multiple arguments, for this we should avoid parentheses bracket.

Eg: (a) echo ("Hello World");
(b) $a=5;
echo "Hello ", $a, (" World");

(III) print_r():
- This function is used to display the structure of value procession as output in user readable format.
- It is used to see the complex variables like array, object etc.

Eg: $arrayVar=array (5, 1, 3, 2, 5, 0);
print_r ($arrayVar);

 # Constants:

- Constants in PHP are defined using define().
- Conventionally, constant name are given in all caps form.
- Once the constants are defined their value cannot be changed later.
- Only scalar data (boolean, integer, double and string) can be contained in constants.

Syn: define(constant_name, value)

Example: define("My_Roll", 13);
define("My_Subject", "PHP");
echo My_Roll . "<br>". My_Subject;
echo constant(My_Roll);

print_r (get_defined_constants()); // returns the names and values of all the constants currently defined.

*** Built-in Constants:**

1. PHP_OS
2. PHP_VERSION
3. E_ERROR
4. WARNING
5. E_PARSE
6. E_NOTICE
7. E_ALL

 # Comments:

(1) Single line comment:
(a) // your text here
(b) # your text here

(2) Multiple line argument:
/*
your text here
*/

 # Data Types:

(1) Integer:
Example: $a=513;

(2)double:
Example: $a=513.250;

(3) String:
Example: $a= “Hello World”;
$b= ‘Web Developer’;

(4)Boolean:
Example: $a= TRUE;

$b= FALSE;

(5)Null:

(6) Array:
Example: $a= array(“foot”, “bike”, “car”, “plane”);
$b=array(5,1,3, 1,9,7);

(7) Object:
Example: $objVar= new ClassName();

(8) Resource:
It is a reference to the external resources outside the PHP such as My Sql server, File handling etc.
Example: $link= mysql_connect(“localhost”, “root”, “ ”);
$fp= fopen(“page.txt”, “r”);

*** Type Testing:**

(1) gettype():
This function returns string representation of data types of the variable i.e. either integer, string, double, Boolean etc.
Example: $a=2468;
$b=gettype($a);
echo ($b); // integer

(2) is_int():
It is a Boolean function that returns true if an argument in integer type, else returns false.
Example: echo is_int(123); //1(True)
Echo is_int(123.456); // (False)
(3) is_integer():
Alias of is_int().

(4) is_long():
Alias of is_int().

(5) is_double():
It is a Boolean function that returns true if an argument is double type, else returns false.

Example: echo is_double(3.1456); // 1(True)
echo is_double("developer"); // (False)

(6) is_float():
Alias of is_double().

(7) is_real():
Alias of is_double().

(8) is_bool():
Example: $a= is_int(12);
echo is_bool($a); //1(True)

(9) is_string():
Example: echo is_string(1234); // (False)
echo is _string("1234"); //1(True)

(10) is_array():
Example: $a= array(5,1,3); //1(True)

(11) is_object():
Example: $a =5;
echo is_object($a); // (False)
$b= new ClassName();
echo is_object($b); //1(True)

(12) is_resource():
Example: $link= mysql_connect("localhost", "root", " ");
echo is_resource($link); //1(True)

*** Type Conversion:**

To change the data type of the variable, we use settype().
Example: $a=123;
echo gettype($a); //integer
settype($a, string);
echo gettype($a); // string

*** Variable Testing:**

To check whether the variable exists or not or its value procession, we use isset().

Isset() is a Boolean function.

Example: echo isset($abc); // (False)
$abc= 1;
echo isset($abc); //1(True)

*** Destroy Variable:**

To destroy variable we use unset().
$abc= 1;
echo isset($abc); //1(True)
unset ($xyz);
echo isset($xyz); // (False)

*** Check Empty:**

To check whether the variable has empty value or not, we use empty().
Following are the list of empty values:
(a) $a= " "; (empty string)
(b) $a= NULL
(c) $a= FALSE
(d) $a=array ();
(e) var$a;
echo empty($a); //1(True)

 # Condition Structure:

(1) if():

Syn: if (condition) {
Statement
}

if(condition):
Statement;
endif;

Example: (a) $a=5;
If ($a==5);{
echo ("value of a variable is 5");
}

Comparison operator
= = Equality
< = less than or equal to
> = greater than or equal to

! = not equal to

Assignment operator
= assign value to variable

Conditional operator
&& And: Both must be true.
|| OR: Anyone is true.

```
(b) if($a>=5){
      echo("value of a variable is greater than 5");
      }
```

Multiple conditions

```
(I) $a=5;
      $b= "Hello";
if ($a==5, && $b= "Hello") {
echo ("Both conditions are true");
}
```

```
(II) $a=5;
      $b= "Hello";
     if ($a==5, || $b== "Hello"){
     echo ("Anyone is true");
    }
```

```
(2) else()
Syn: if (condition){
       Statement-1;
      }
else{
statement-2;
}
```

```
Example: (I)$a=6;
            If ($a> 10){
echo ("variable is greater than 10");
}
else{
echo ("variable is less than 10");
}                    // variable is less than 10.
```

(II) $a=6;
if (is_int($a)){
echo (“variable is integer”);
}
else{
echo (“variable is not integer”);
} // variable is integer

(3) elseif:
Syn: if (condition1) {
Statement-1;
}
elseif{
statement-2;
}
Example: $a=6;
If ($a>10) {
echo (“variable is greater than 10”);
}
elseif{
echo (“variable is less than 10”);
}
elseif{
echo (“variable is equal to 10”);
}

 # Looping Structure:

(1) while():
Syn: while(condition){
Statement1;
}

Example: (i)$a=1;
while($a<=10){
echo "count is :" .$a . "<br>"; $a++;

(ii))$a=1;
while($a>=10){
echo "count is :" .$a . "<br>"; $a--;

(2) do….while()
Syn: do{
Statement;

```
        }
        while (condition);
Example: )$a=1;
         do{
        echo "count is :" .$a . "<br>";  $a++;
        }
        while($a<=10);
```

(3) for():

Syn: for(initialization, condition, loop end){
Statement;
}

```
Example: for($a=1; $a<=10; $a++){
         echo "count is: " .$a. "<br>";
          }
```

(4) foreach():

Later discussed in array.

 # Array:

An array is a list that holds multiple values called as element referenced by index that is either numeric or string.

Initializing array:

(i) directly assigning values to an array:

```
Example: $lang[]="French";
         $lang[]="English";
         $lang[]="German";
         $lang[]="Spanish";
         echo $lang[0]; // French
```

(ii) To explicitly specify an index:

```
Example: (a) $lang[0]="French";
         $lang[1]="English";
         $lang[2]="German";
         $lang[3]="Spanish";
         echo $lang[0]; //French

        (b) $lang[0]="French";
            $lang[2]="English";
            $lang[5]="German";
            $lang[4]="Spanish";
            echo $lang[5]; //German
```

(c) $lang[0]="French";
$lang[3]="English";
$lang[5]="German";
$lang[]="Spanish";
echo $lang[6]; //Spanish

(iii) Using array() construct:
Example: $lang=array("french","english","german","spanish");
echo $lang[3]; //spanish

(iv) To explicitly specify an index array() construct:
Example: $lang=array ("0"=>"french", "1"=>"english", "2"=>"german",
"3"=>"spanish");

(v) Array indices may also be string.
Example: (a)$lang=array ("a"=>"french", "b"=>"english", "c"=>"german",
"d"=>"spanish");
echo $lang["a"]; //french

(b) $lang=array ("bat"=>"french", "ball"=>"english",
"stump"=>"german", "wicket"=>"spanish");
echo $lang["ball"]; //english

 # Looping through array

To perform loop through an array we use typical foreach() loop.

(a) foreach(array as value){
statement;
}
Example: foreach($lang as $val){
echo "language is: " .$val. "<br>" ;
}

(b) foreach(array as key=> value){
statement ;
}
Example: foreach($lang as $key=>$val){
echo "Element" .$key. "has value:" .$val. "<br>" ;
}

*** Built-in Array Functions:**

(1) is_array():
It is a Boolean function that returns true if an argument is array type.
Example: $lang= array("French", "English", "German", "Spanish");
echo is_array($lang);

(2) sizeof():
This function returns numeric size of an array.
Example: $lang= array("French", "English", "German", "Spanish");
echo sizeof($lang);

(3)count():
Alias of sizeof().

(4) in_array()
It is a Boolean function that returns true if an argument value exists as an array value.
Example: $lang= array("French", "English", "German", "Spanish");
echo in_array($lang);

(5) current():
Every array has an internal array point to its current element, which is initialized to the first element inserted into the array. The current() simply returns the value of an array element that is currently being pointed to by the internal array pointer. If the internal array pointer points beyond the end of element list, current function returns false.
Example: $transport= array("foot", "bike", "car", "plane");
echo current($transport);

(6) next():
This function advances the internal array pointer one place forward and returns the element value i.e. currently being pointed to by the internal array pointer.
Example: $transport= array("foot", "bike", "car", "plane");
echo next($transport);
It returns false if next function points beyond element list.

(7) prev():
This function rewinds the internal array pointer one place backward and returns element value i.e. currently being pointed by the internal array pointer. It returns false if prev() points before the element list.

Example: $transport= array("foot", "bike", "car", "plane");
echo prev($transport); //foot

(8) end():

This function advances internal array pointer to the last element of an array and returns the element value i.e. currently being pointed to by the internal array pointer.

```
Example:  $transport= array("foot", "bike", "car", "plane");
          echo end($transport);  //plane
```

(9) reset():

This function rewinds the internal array pointer to the first element of an array and returns element value i.e. currently being pointed to by the internal array pointer.

```
Example: $transport= array("foot", "bike", "car", "plane");
         echo reset($transport); //foot
```

(10) pos():

Alias of current()

(11) key():

This function returns a key of a current array pointer position pointed by the internal array pointer.

```
Example: $transport= array("foot", "bike", "car", "plane");
         echo key($transport);  // 0
```

(12) list():

This function is used to assign array values to that many variables.

```
Example: list($fname, $mname, $lname)= array("Nabin", "k", "Bhattarai");
         echo($fname);
         echo ($mname);
         echo ($lname); // Nabin k Bhattarai
```

(13) unset():

```
Example: $transport= array("foot", "bike", "car", "plane");
         print_r($transport);
         unset ($transport[2]);
         print_r ($transport);
```

(14) array_push():

This function is used to insert value on to the end of an array.

```
Example: $transport= array("foot", "bike", "car", "plane");
         array_push($transport, "train", "ship");
         print_r($transport);
```

(15) array_unshift(): This function is used to insert value on to the begining of an array.

(16) array_pop():

This function pops and returns the last value of the array, shortening the array size by one element.

```
Example: $transport= array("foot", "bike", "car", "plane");
         $val= array_pop($transport);
         print_r($transport);
         echo $val; //ship
```

(17) array_shift():

This function pops and returns the first value of the array, shortening the array size by one element.

(18) array_merge():

This function merges the element of one or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array.

Example:
```
$lang= array("French", "English", "German", "Spanish");
$transport= array("foot", "bike", "car", "plane");
$new= array_merge($lang, $transport);
print_r ($new);
```

(19) sort():

This function sorts and array. Elements will be arranged from the lowest to highest, if numerically and alphabetically.

Example:
```
$lang= array("French", "English", "German", "Spanish");
sort($lang);
print_r($lang);
```

(20) shuffle():

This function randomizes the order of elements in an array.

Example:
```
$lang= array("French", "English", "German", "Spanish");
shuffle($lang);
print_r($lang);
```

(21) ksort():

This function sorts and array by key maintaining key to data correlation.

Example:
```
$fruits= array("c"=> "lemon", "a"=> "mango", "d"=> "apple", "b"=>
"banana");
ksort ($fruits);
print_r($fruits);
```

(22) explode():

This function returns an array of strings each of which is a sub-string of a string.

Example:
```
(a) $arrstring= explode("", "This is a test");
    print_r($arrstring);
(b) $arrstring=explode("s", "This is a test");
    print_r ($arrstring);
```

(23)implode():

This function returns a string containing a string representation of all the array elements in the same order.

Example:
```
(a) $str= implode("", "This is a test");
    print_r($str);
(b) $str= implode(",", "This is a test");
    print_r($str);
```

(24) array_reverse():
This function returns the new array reversing the order of elements.
Example: $lang= array(“French”, “English”, “German”, “Spanish”);
$newlang= array_reverse($lang);
print_r($newlang);

(25) array_rand():
This function picks one or more random entries out of an array.
Example: $transport= array(“foot”, “bike”, “car”, “plane”);
$newtransport= array_rand($transport, 2);
print_r($newtransport);
echo $transport [$newtransport[0]];

(26) array_sum():
This function returns the sum of values in an array as an integer or float.
Example: $num= array(2, 4, 6, 8);
$sum=array_sum($num);
echo $sum;

(27) array_shift():
This function shifts an element off the beginning of an array.
Example: $transport= array(“foot”, “bike”, “car”, “plane”);
$val= array_shift($transport);
print_r ($transport);
echo $val;

(28) array_unshift():
This function adds one or more elements to the beginning of an array.
Example: $transport= array(“foot”, “bike”, “car”, “plane”);
array_unshift($transport, “train”);
print_r($transport);

Function:

A function is a block of code that is defined in one location and then is invoked from any other part of the program.
There are two types of functions:

(a) Built-in function: It is already defined in PHP.
Example: gettype(), settype(), is_int(), print() etc.

PHP has numerous built in function.

(b) User-defined function: This is a function defined by the programmer himself.
To define a function we use "function" keyword.

Syn: function function_name(arg1, arg2, arg3,……..argN){
Statement-1;
Statement-2;
Statement-3;

Statement-N;
return value;
}

Some examples:

(i) Calculating the area

```
function calculateArea($l,$b){
$a=$l*$b;
return $a;
}
$area=calculateArea(5,3);
echo "Area:".$area;
```

(ii) Calculating the factorial

```
function calculateFactorial($n){
if ($n==0  || $n==1){
return 1;
  }
else{
$fact=1;
for ($i=1; $i<=$n; $i++){
$fact= $fact * $i;
}
```

```
return $fact;
}
}
echo calculateFactorial(5);
// 120
```

(c) Calculating the sum of odd numbers between 1 and 100

```
function sumofodd($a,$b){
$sum =0;
for ($i=$a; $i<=$b; $i=$i+2){
$sum=$sum+$i;
}
return $sum;
}
echo sumofodd(1,100);
```

*** Function and Variable Scope:**

By default, variables inside function have local scope. Variables outside the function have global scope.

Example:
(1) Local:

```
function printAge(){
$age=20;
echo $age;
}
$age=30;
echo $age; //30
print Age(); //20
echo $age; //30
```

(2) Global:

```
 function printAge(){
global $age;          //$GLOBALS[“AGE”]=20;
$age=20;
echo $age;            //echo $GLOBALS[“AGE”];
}
$age=30;
echo $age; //30
print Age(); //20
echo $age; //20
```

*** Built-in functions for function:**

(i) func_num_args():
This function returns the number of argument passed to the function.
Example:
```
function argTest($a, $b, $c, $d){
    echo func_num_args(); //14
    }
    argTest ("w", "x", "y", "z");
```

(ii) func_get_arg(index):
This function takes index as argument and returns argument from the argument list according to index.
Example:
```
function argTest($a, $b, $c, $d){
    echo func_get_args(); // 4
    echo func_get_arg(2); //y
      }
    argTest("w", "x", "y", "z");
```

(iii) func_get_args():
This function returns array containing all the functions argument.
Example:
```
function argTest($a, $b, $c, $d){
    echo func_num_args(); //4
    echo func_get_arg(2); //y
    $arrayArg= func_get_args();
    print_r ($arrayArg);
    }
    argTest("w", "x", "y", "z");
```

 # STRING:

(A) Comparison and Searching:

(1) strlen():
This function returns numeric length of a string.
Example: $text="This is a test";
echo strlen($text); //14

(2) strpos():
This function returns numeric position of the first occurrence of a needle in
the string else returns false if needle is not found in the string.
Syn: int strops(string, needle)
Example: $text= “This is a test”;
echo strpos($text, “s”); //3
echo strops($text, “is”); //2

(3) strrpos():
This function returns the numeric position of the last occurrence of a needle in the string.
Syn: int strrpos(string, needle)
Example: $text= “This is a test”;
echo strrpos($text, “s”); //12
echo strrpos($text, “is”); //5

(4) strstr():
This function returns sub-string from the string starting with the needle else returns false.
Syn: string strstr(string, needle)
Example: $text= “user@example.com”;
echo strstr($text, “e”); //er@example.com

This function is case sensitive.
Example: $text= “usEr@example.com”;
echo strstr($text, “e”); //example.com

(5) strchr(): alias of strstr()

(6) stristr():
This function is similar to strstr() except it is case insensitive.
Syn: string stristr(string, needle)
Example: $text= “usEr@example.com”;
echo stristr($text, “e”); //Er@example.com

(7) strcmp():

This function compares two string, string1 and string2 according to its ASCII(American Standard Code for Information Interchange) code and returns the following values.

0 => if string1 = string2
1=> if string 1> string2
-1=> if string1< string2
Syn: strcmp(string1, string2)
Example: echo strcmp("abc", "abc"); //0
echo strcmp("Abc", "abc"); //-1
echo strcmp("abcg", "abcd"); //1

Exceptional case: echo strcmp("abc", "abcdef"); //-3
echo strcmp("abc", "bbcdef"); //-1

(8) strcasecmp():

This function is similar to strcmp() except it is case insensitive.

Example: echo strcasecmp("abc", "abc"); //0
echo strcasecmp("Abc", "abc"); //0

Exceptional case:
echo strcasecmp("abcg", "abcd"); //3
echo strcasecmp("abc", "abcdef"); //-3
echo strcasecmp ("abc", "abcdef"); //-1

(B) <u>Substring Selection</u>:

(9) substr():

Syn: substr(string, start, length)
This function returns a sub-string from the string according to the start and length value.

(a) If start is non-negative, the returned string will start at the start position counting from zero.

Example: $n= "abcdefgh";
echo substr($n, 3); //defgh

(b) If length is given and is positive, the string returned will contain at most length characters beginning from the start.

Example: : $n= "abcdefgh";
echo substr($n, 2, 3); //cde

(c) If start is negative, the returned string will start at the first character from the end of the string.

Example: : $n= “abcdefgh”;
```
echo substr($n, -3); //fgh
echo substr($n, -6, 4); //cdef
```

(d) If the length is negative, then that many characters will be omitted from the end of the sub-string.

Example: : $n= “abcdefgh”;
```
echo substr($n, 2, -3); //cde
echo substr($n, -5, -2); //def
echo substr($n, 5, -5);
echo substr($n, -2, -3);
```

(C) <u>String Clean-up</u>

(10) trim():

This function trims following default characters in string from both left and right end. This function also trims user-supplied characters from left and right end.

“ ”(white space)
\ n (new line)
\ t (tab space)
\ r (carriage return)

Syn: string trim(string, char list)

Example: $text="Hello World";
```
echo trim($text,"Hld"); //ello Wor
```

(11) ltrim():

This function is similar to trim(). It trims only from left side.

(12) rtrim():

This function is similar to trim(). It trims only from right side.

(13) chop(): alias of rtrim()

(D) String replacement:

(14) str_replace():
This function replaces sub-string in the string by new sub-string to return new-string.
Syn: string str_replace(substr1,substr2, string)
Example: (a) echo str_replace("This", "That", "This is a test"); //That is a test
(b) $phrase= “You should eat fruits, vegetables and fiber”;
$healthy= array(“fruits”, “vegetables”, “fiber”);
$yummy= array(“pizza”, “momo”, “ice-cream”);
$newphrase= str_replace($healthy, $yummy, $phrase);
echo $newphrase;

(15) ereg_replace():
similar to str_replace.

(16) substr_replace():
This function replaces certain number of characters by new characters according to start, end and length value.
Syn: substr_replace(string, substr, start, length)
Example: echo substr_replace("abcdefgh", " ", 2,3); //ab---fgh

(17) substr_count():
This function returns the number of occurrence of needle in the string.
Syn: int substr_count(string, needle)
Example: $text="This is a test";
echo substr_count($text, "s"); //3
echo substr_count($text, "is"); //2

(18) str_repeat():
This function returns new string repeating original string according to repetition value.
Syn: str_repeat(string, repetition_value)
Example: echo str_repeat("cheers ", 3); // cheers cheers cheers

(19) strrev():
This function returns the reverse of a string.

Example: echo strrev("developer"); //repoleved

(20) chr():
This function takes ASCII code of character as argument and returns actual character.
Example: echo chr(65); //A

(21) ord():
This function takes actual character and returns respective ASCII code.
Example: echo ord(a); //97

(E) Case Function:

(20) strtolower():
This function converts the case of string to lower.
Example: echo ucwords("ThIs Is A tEsT"); // this is a test

(21) This function converts the case of string to upper.
Example: echo ucwords("ThIs Is A tEsT"); // THIS IS A TEST

(22) This function converts the case of first character of string to upper.
Example: echo ucwords("ThIs Is A tEsT"); // This is a test

(23) ucwords():
This function converts the case of first character of each word to upper.
Example: echo ucwords("This is a test"); // This Is A Test

(F) Escaping function:

(24) addslashes():
This function inserts slash(\) as escape sequence before following characters.
' (Single Quote)
" (Double Quote)
\ (Back Slash)

```
Example: echo addslashes("Author is O' Reilly");
         // Author is O\' Reilly
         $text= "Ram is a  \"good\"  boy";
         echo $text;
         // Ram is a "good" boy
```

(25) stripslashes():
This function removes slashes used for the purpose of escaping sequences.

Example: echo stripslashes("Author is O\'Reilly");
// Author is O'Reilly

(G) HTML Functions:

(26) htmlspecialchars():
This function converts following special characters to HTML entities. In other words, it prints HTML tags as it is.
'(single quote)
"(double quote)
<(less than sign)
>(greater than sign)
&(Ampersand)

Example:

```
$text="<b>Registered</b> users, please <a href=\"login.php\"
click</a>to <i>login</i>";
echo $text;
```

(27) htmlentities():
similar to htmlspecialchars()

(28) strip_tags():
This function omits HTML tags in the string. Tags supplied as allowed tags are not omitted.
Syn: string strip_tags(strings allowed_tags);
Example:

```
$text="<b>Registered</b> users, please <a href=\"login.php\"
        click</a>to <i>login</i>";
        echo strip_tags($text);
```

(29) nlb2r():
This function inserts HTML break in the string.
Example:

```
$text="This
    is
    a
    test";
    echo $text;
    echo nl2br($text);
```

 # PHP and MYSQL Functions:

1) mysql_connect():
This function is used to connect to MySQL server using specified hostname, username and password.
Syn:
resource mysql_connect(string hostname,string username, string password)
Ex:
$link = mysql_connect("localhost","root","") or die("Could not connect to MySQL server");

2) mysql_select_db():
This function is used to select database from MySQL server.
Syn:
int mysql_select_db(string database,resource link)
Ex:
$db = mysql_select_db("db_profile",$link) or die("Could not find database from MySQL server");

3) mysql_query():
This function is used to send SQL statement to MySQL server to be executed. This function returns TRUE for INSERT,UPDATE,DELETE SQL statements, but resource type resultset for SELECT SQL statement.
Ex:

```
$query = "INSERT INTO tbl_profile VALUES ('','Bikash','abc','11000','M')";
mysql_query($query) or die(mysql_error());
```

4) mysql_affected_rows():
This function returns the number of rows affected after using recent INSERT, UPDATE, DELETE SQL statements.

```
$affRows = mysql_affected_rows();
if($affRows==1){
        echo $affRows."  record inserted successfully";
}
else{
        echo "Error while inserting data";
}
```

5) mysql_error():
This function is used to display the error generated by the mysql functions.

6) mysql_fetch_result:
This function returns the result from the intersection of the rows and columns.
Syn: mysql_fetch_result(resource resultset,int rows, int cols)
Ex:

```
$sql = "SELECT * FROM tbl_profile";
```

```
$result = mysql_query($sql);
$name = mysql_fetch_result($result,2,1);
echo $name; //Sita
```

7)mysql_fetch_array():

This function returns an array of single record from the resultset.We fetch those values according to the following values MYSQL_NUM, MYSQL_ASSOC, and MYSQL_BOTH(default).

a)MYSQL_NUM:Returns an enumerated array.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$array = mysql_fetch_array($result,MYSQL_NUM);
echo "Name: ".$array[1]."<br>";
echo "Address: ".$array[2]."<br>";
```

b)MYSQL_ASSOC: Returns an associative array.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$array = mysql_fetch_array($result,MYSQL_ASSOC);
echo "Name: ".$array['profile_name']."<br>";
echo "Address: ".$array['profile_address']."<br>";
```

c)MYSQL_BOTH: Returns an associative array.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$array = mysql_fetch_array($result,MYSQL_BOTH);
echo "Name: ".$array[1]."<br>";
echo "Address: ".$array['profile_address']."<br>";
```

d)

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
while($array = mysql_fetch_array($result)){
  echo "Name: ".$array['profile_name']."<br>";
  echo "Address: ".$array['profile_address']."<br>";
}
```

8)mysql_fetch_row():

This function fetches the resultset as an enumerated array.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
```

```
$row = mysql_fetch_row($result);
echo "Name: ".$array[1]."<br>";
echo "Address: ".$array[2]."<br>";
```

9)mysql_fetch_object():

This function fetches the resultset as an object.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$object = mysql_fetch_object($result);
echo "Name: ".$object->profile_name."<br>";
echo "Address: ".$object->profile_address."<br>";
```

10) mysql_num_rows():

This function returns the number of rows returned in the resultset.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$numRows = mysql_num_rows($result);
echo $numRows." record(s) fetched successfully";
```

11) mysql_num_fields():

This function returns the number of columns fetched in the resultset.

12) mysql_field_name():

This function returns the name of the fields fetches in the resultset.

Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$numFields = mysql_num_fields($result);
for($i=0; $i<$numFields; $i++){
   $fieldName = mysql_field_name($result);
   echo "Field Name: ".$fieldName."<br>";
}
```

13) mysql_insert_id():

This function returns the AUTO_INCREMENT generated ID from the record inserted by the latest INSERT SQL statement.

If the ID is not AUTO_INCREMENT then this function returns 0(zero).

```
$sql = "INSERT INTO tbl_profile VALUES
            ('','Bikram','Buddhanagar','18000','M')";
$insert = mysql_query($sql);
$pid = mysql_insert_id();
echo $pid;
```

14) mysql_data_seek():

This function moves internal row pointer in the resultset starting from zero.
Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
mysql_data_seek($result,3);
$array = mysql_fetch_array($result);
echo "Name: ".$array[1]."<br>";
echo "Address: ".$array['profile_address']."<br>";
```

15) mysql_fetch_aSSOC():
This is similar to mysql_fetch_array() with MYSQL_ASSOC.
Ex:

```
$sql = "SELECT * FROM tbl_profile";
$result = mysql_query($sql);
$array = mysql_fetch_assoc($result);
echo "Name: ".$array['profile_name']."<br>";
echo "Address: ".$array['profile_address']."<br>";
```

16) mysql_close():
This function is used to close the connection with the MySQL server.
Ex: mysql_close($link);

File Inclusion in PHP:

1) include():
-This function is used to include external files to the working page.
-This function generates warning if error occured while inclusion, like file not found.
Ex: include('test.php');

2) include_once():
-This function checks whether given file is already included. If included doesnt include again if not includes the file.
-This function generates warning if error occured while inclusion, like file not found.
-This function avoids the problem of redefine.
Ex: include_once('test.php');

3) require():
-This function is used to include external files to the working page.
-This function generates fatal error if error occured while inclusion, like file not found.
Ex: require('test.php');

4) require_once():
-This function checks whether given file is already included. If included doesnt include again if not includes the file.
-This function generates fatal error if error occured while inclusion, like file not found.
-This function avoids the problem of redefine.
Ex: require_once('test.php');

Session:

Session are stored in the web server. Session variables are need not to be passed from one page to another.

* Configure Session:
1) Open the folder c:\phpdev5\gtkdev\php4\
2) From which copy the file "php.ini" to system folder i.e. c:\winnt\
3) Open that copied file php.ini
4) Search the settings session.save_path = /tmp
5) Create the folder "temp" inside c:\phpdev5\www\ folder.
6) Now change the value to session.save_path = c:/phpdev5/www/temp/ and save the file.
7) Restart apache server if already started.

* Built-in function for Session:
1) session_start():
-This function starts the session.
-To work with any session variables or function we should start the session.
-This function should be called at very top of the page before any php script or html tags.

2) session_register():
-This function registers the given variable string to the current session.
-This function starts the session if the session is not started before.
Ex:
```
$os = "windows";
session_register("os");
echo $_SESSION['os'];        //windows
```

3) session_is_registered():
-It is a boolean function that returns true if the given variable string is registered to the current session.
Ex:
```
//session_start();
echo session_is_registered("os"); //1
echo isset($_SESSION['os']); //1
```

4) session_unregister():
This function deletes the given session variable from the current session.
Ex:
```
session_unregister("os");
unset($_SESSION['os']);
```

5) session_name():
-When this function is called with no argument, it returns the current session name.
-When called with argument string it assigns the current session name to given string.
Ex:
```
//session_start();
```

```
echo session_name();//PHPSESSID
session_name("my_session");
echo session_name();//my_session
```

6) session_id():
-When this function is called with no argument, it returns the current session id, which is the unique string to identify current session.
-It is the 32-char long alphanumeric string.
-When called with argument string it assigns the current session id to given string.

```
Ex:     //session_start();
        echo session_id();    //40f9d43e4ea232e6dbe633675a44cf14
        session_id("my62_session0927596");
        echo session_id();    //my62_session0927596
```

7) session_unset():
This function is used to destroy all the session variable from the current session.

```
Ex:     //session_start();
        session_unset();
```

8) session_destroy():
This function detroys the current session.

```
Ex:     //session_start();
        session_destroy();
```

9) session_cache_expire(): (supported in >=php4.2)
-This function is used to set the life span of the current session, when to expire.
-Default is 80 minutes.

```
Ex:     //session_start();
        session_cache_expire(30);    //expires after 30 min.
```

* Application:
1) For super global use.
2) Login validation.
3) User authentication
4) Registeration process enhancement.
5) Hit counter
6) Shopping cart.

*Counter

```
<?
        session_start();
        if(isset($_SESSION['counter'])){
                $_SESSION['counter']++;
        }
        else{
                $_SESSION['counter']=1;
        }
        echo $_SESSION['counter'];

?>
```

 # Cookie:

-Cookies are a mechanism for storing data in the remote browser(in client) and thus tracking or identifying return users(by the web server).
-Cookies are sent along with the header information.
-While creating cookies PHP script checks for the previously created cookie file, so as to use that cookie if not exists creates the new one.
-Cookie variables are also the super global variables.

*** Defining cookie:**
setcookie():
This function is used to define a cookie to be sent with the header information.
Syn:
int setcookie(string cookiename, string value, int expire, string path, string domain, int secure);

(a) cookiename: name of the cookie
(b) value: The value of the cookie. This value is stored on the clients computer.
(c) expire: The time the cookie expires.
- set to 0(zero) //cookie expires when browser is closed
- time()+3600 //cookie expires after 1 hr
- mktime(16,24,00,6,20,2006) //expires at 4:24pm 20 June 2006.

(d) path: The path on the server in which the cookie will be available on.
If set to '/', the cookie will be available within the entire domain.
(e) domain: The domain that the cookie is available.
- www.example.com //
(f) secure: Indicates that the cookie should only be transmitted over a secure HTTPS connection.
- When set to 1, the cookie will only be set if a secure connection exists.
- When set to 0(zero), the cookie will be set in any connection

Ex:

```
setcookie("username","prashish",time()+(60*60*24),"/","www.example.com",0);
```

***Destroy Cookies:**
1) Calling the setcookie function with only first argument and no further arguments.
Ex: setcookie("username");

2) Setting the passed expiry time.
Ex: setcookie("username","prashish",time()-(3600));

*** Application:**
1) For super global use.
2) Login validation.

3) User authentication
4) Registeration process enhancement.
5) Hit counter
6) Shopping cart.
7) Remember me feature in login.

***Counter:**

```
if(isset($_COOKIE['counter'])){
        $count=$_COOKIE['counter'];
        $count++;
        setcookie("counter",$count,time()+3600);
}
else{
        $count=1;
        setcookie("counter",$count,time()+3600);
}
echo $_COOKIE['counter'];
```

 # Sending Email:

mail():

-This function is used to send email of plain text or html format.

-Syn: boolean mail(string mail_to,string mail_subject,string mail_body, string extra_headers);

-eg:

(a) mail("bikash@hotmail.com","hi there","This is my test mail. Reply me asap");

(b)
```
$mail_to = "someone@abc.com";
$mail_from = "somebody@xyz.com";
$mail_reply_to = "somebody@xyz.com";
$mail_cc = "anyone1@mno.com,anyone2@mno.com";
$mail_bcc = "otherone1@pqr.com,otherone2@pqr.com";
$mail_subject = "hi frens";
$mail_body = "Hello frens.<br>This is my first PHP test mail.<br>So, please send me <b>feedback</b>";
$mail_headers = "MIME-Version: 1.0\r\n
                 Content-type: text/html; charset=iso-8859-1\r\n
                 From: $mail_from\r\n
                 Reply-to: $mail_reply_to\r\n
                 Cc: $mail_cc\r\n
                 Bcc: $mail_bcc\r\n
                 Subject: $mail_subject\r\n
                 Date: ".date('r')."\r\n";
if(@mail($mail_to,$mail_subject,$mail_body,$mail_headers)){
   echo("Email sent successfully");
}
else{
   echo("Error occured while sending mail");
}
```
//@ - Error message suppressor

File Handling:

1) fopen():

This function is used to open the file in various modes.

Modes	Description
r	Open file in readonly mode.
r+	Open file in both read and write mode.
w	Open file in write only mode. If file doesnt exists then creates the file. If exists then write the content into the file, truncating it , from beginning.
w+	Open file in both read and write mode. If file doesnt exists then creates the file. If exists then write the content into the file, truncating it , from beginning.
a	Open file in write only mode. If file doesnt exists then creates the file. If exists then write the content into the file from the last.
a+	Open file in both read and write mode. If file doesnt exists then creates the file. If exists then write the content into the file from the last.

Syn: resource fopen(string filename,string mode)

Ex:
```
$fp = fopen("test.txt","r");
```

2) fread():

This function is used read the content of file.

Ex:
```
$fp = fopen("test.txt","r");
$buffer = fread($fp,filesize("test.txt"));
echo $buffer;
```

3) fgets():

This function is similer to fread().

4) fwrite():

This function is used to write the string into the file.

Ex:
```
$fp = fopen("test.txt","w");
$string = "This is a test";
//$string = "This\nis\na\ntest";
$size = fwrite($fp,$string);
```

5) feof():

This boolean function returns true if the file pointer is at the end of file.

Ex:
```
$fp = fopen("test.txt","r");
if($fp){
                    while(!feof($fp)){
```

```
                $buffer = fread($fp,1024);
                echo $buffer;
                }
                fclose($fp);
        }
```

6) file():
This function is used to read the content of file and returns an array such that each line corresponding.
Ex: $array = file("test.txt");
print_r($array);

7) fclose():
This function is used to release the memory associates the file pointer.
Ex: fclose($fp);

8) file_exists():
This is a boolean function that returns true if the given file exists.
Ex: echo file_exists("test.txt");

9) file_type():
This function returns the type of an item whether it is file, folder, etc.
Ex: echo file_type("test.txt"); //file

10) copy():

11) unlink():
This function deletes the file permanently.
Ex: unlink("test.txt");

12) rename():
This function is used to rename the file or folder.
Ex: rename("test.txt","newtest.txt");

13) is_dir():
It is a boolean function that returns true if the given string is folder name.
Ex: echo is_dir("images");

14) mkdir():
This function is used to make the folder/directory.
Ex: mkdir("images",0777);

15) rmdir():
This function is used to remove the folder/directory.
Ex: rmdir("images");

16) opendir():
This function is used to open the directory and returns the folder handle.

17) readdir():
This function is used to read the items contained by the folder.

18) closedir():
This function released the folder handle associated memory.

Ex:
```
if(is_dir("images")){
    $fd = opendir("images");
    while($item = readdir($fd)){
        echo $item;
    }
    closedir($fd);
}
```

#Date/time function in PHP

***date():**

This function returns the current(running) date and time string according to format character.

Format Char	*Description*
d	Numeric representation of day of a month with leading zero.(01 to 31) eg: echo date('d'); //04
D	Short string representation of day of a week.(Mon to Sun)
j	Numeric representation of day of a month without leading zero.(1 to 31)
l	Full string representation of day of a week.(Sunday to Saturday)
w	Numeric representation of day of week.(0(Sunday) to 6(Saturday))
z	Day of a year starting from 0(0 to 365)
Week:	
W	Week no of a year, each week starts from Monday.(0 to 52)
Month:	
F	Full string representation of a month.(January to December)
m	Numeric representation of month with leading zero.(01 to 12)
M	Short string representation of month(Jan to Dec)
n	Numeric representation of month without leading zero(1 to 12)
t	No. of days in given month.(28 to 31)
Year:	
L	Check wether it is a leap year(1-Leap year / 0-No)
Y	Full numeric representation of year(2006)
y	Short numeric representation of year(06)
Time:	
a	Lowercase Antemeridiem/Postmeridiem (am/pm)
A	Uppercase Antemeridiem/Postmeridiem (AM/PM)
g	12-hr format of hour without leading zero(1 to 12)
h	12-hr format of hour with leading zero(01 to 12)
G	24-hr format of hour without leading zero(0 to 23)
H	24-hr format of hour with leading zero(00 to 23)
i	Minute with leading zero(00 to 59)
s	Seconds with leading zero(oo to 59)
Timezone:	
O	Diff to GMT in hours.(+52700)
Full Date/Time	
r	Formatted date(Sun, 12 June 2006 10:37:00 +52700)

Inserting into database

Datatype:
1)date (2006-06-05)
$date = date('Y-m-d');

2)time (09:48:43)
$time = date('H:i:s);

3)datetime (2006-06-05 09:51:34)
$now = date('Y-m-d H:i:s');

***time():**
This function returns current time measured in no of seconds since January 1 1970 00:00:00 GMT.
eg: echo time(); //465465475

***mktime():**
This function is use to make datetime string.
Syn: string mktime(Hour,Minute,Second,Month,Day,Year);

```
//Jun 4,2006 10:03am
$date = $row['added_date'];
$array = explode(' ',$date);
$array1 = explode('-',$array[0]);
$array2 = explode(':',$array[1]);

echo date("M j,Y g:ia",
mktime($array2[0],$array2[1],$array2[2],$array1[1],$array1[2],$array1[0]));
```

 # Object Oriented Programming:

***Class:**

It is the collection of variables and functions, termed as member variables or member functions. To access class we create an object.

Keyword 'class' is used to define class.

Syn:

```
class ClassName{
        var $memvar1;
        var $memvar2;
        ...
        var $memvarN;

        //contructor
        function ClassName(){
          //body
        }
        function method1(){
          //body
        }
        function method2(){
          //body
        }
        ...
        function methodN(){
          //body
        }
}//end of class
```

Eg:

```
class TextBoxSimple{
        var $body_text = "my text";
        function display(){
           echo('<table border="1"><tr><td>');
           echo $this->body_text;
           echo('</td></tr></table>');
        }
    }
```

***Object:**

It is an instance of a class ie used to access the particular class.

```
eg:     $objBox = new TextBoxSimple();
        $objBox->display();
        $objBox->body_text = "custom text";
        $objBox->display();
```

***Constructor:**

It is a special function that shares the name of class and is invoked when an object is created.

Eg:

```
class TextBox{
        var $body_text = "my text";
        function TextBox($body_text_in){
           $this->body_text = $body_text_in;
        }
        function display(){
           echo('<table border="1"><tr><td>');
           echo $this->body_text;
           echo('</td></tr></table>');
        }
}
$objBox = new TextBox("Custom text");
$objBox->display();
```

***Inheritence:**

-Process of creating new class from the existing class.
-Class which is derived is called base class, parent class or super class.
-Deriving class is called child class or subclass.
-Automatically child class has all the member variables and member functions as the parent class.
-Multiple inheritence is supported in PHP.
-Inheritence in PHP is done using the 'extends' clause.

Syn:

```
class Child extends Parent{
   //body
}
```

eg:

```
class TextBox{
        var $body_text = "my text";
        function TextBox($body_text_in){
           $this->body_text = $body_text_in;
        }
        function display(){
           echo('<table border="1"><tr><td>');
           echo $this->body_text;
           echo('</td></tr></table>');
        }
}
class TextBoxHeader extends TextBox{
   var $header_text;
   //constructor
```

```
    function TextBoxHeader($header_text_in,$body_text_in){
            $this->header_text = $header_text_in;
            $this->body_text = $body_text_in;
    }
    function display(){
            echo('<table border="1"><tr><td>');
            echo($header_text);
            echo('</td></tr><tr><td>');
            echo($body_text);
            echo('</td></tr></table>');
    }
}

//Parent class
$objNewBox = new TextBox("Custom text");
$objNewBox->display();
//Child class
$objChildBox = new TextBox("Custom text1","Custom text2");
$objChildBox->display();
```

```
<?      //clsDbconn.php
        class Dbconn{
                var $host;
                var $uname;
                var $psw;
                var $dbase;
                var $link;
                var $db;

                //Constructor
                function Dbconn(){
                        $this->host="localhost";
                        $this->uname="root";
                        $this->psw="";
                        $this->link = mysql_connect($this->link,$this-
>uname,$this->psw) or die("Couldnot connect to MySQL server");
                        $this->db = mysql_select_db($this->dbase,$this->link) or
die("Couldnot find database");
                }
                function Dbclose(){
                        mysql_close($this->link);
        }
                                }
        }//end of class      ?>
```