



# PHP & MySQL with Project

---

## Table of Contents

Overview of PHP .....	5
Advantage of PHP .....	5
PHP Syntax .....	5
Steps to run PHP .....	6
PHP Printing .....	6
PHP Variable .....	7
PHP Concatenation .....	7
Constant in PHP .....	7
PHP Data Types .....	8
PHP Comments .....	10
if else Statement .....	10
if elseif Statement .....	11
switch Statement .....	11
Loop .....	12
for Loop.....	12
while Loop.....	12
do while Loop .....	13
foreach Loop .....	13
break Statement .....	14
continue Statement .....	14
Array .....	15
Array Handling Function .....	17

Function .....	17
General Function .....	21
String Function .....	23
Date Function .....	25
Math Function .....	26
PHP Super globals .....	27
Cookie .....	31
Session .....	34
File Handling .....	37
File Handling Function .....	37
File Opening Mode.....	37
Create User .....	40
Drop User.....	41
Create Database .....	41
Create Table.....	41
Drop Table .....	41
Select Statement.....	41
Comparison Operator .....	43
Pattern Matching.....	43
Logical Operator .....	44
Order by.....	44
String Function.....	45
Number Function.....	46

Date Function .....	46
Data Type Conversion .....	47
Date Formats .....	47
Aggregate Functions .....	48
Group by and Having clause.....	48
MySQL Join .....	49
Database Programming using PHP .....	49
AJAX Concepts .....	56
PHP Filters.....	59
Error Handling.....	60
OOPS Concepts .....	62
Class and Object Concepts .....	62
\$this Keyword .....	64
Constructor and destructor in PHP .....	65
Access Modifiers .....	68
Inheritance.....	70
Method Overriding .....	72
Encapsulation and Abstraction .....	73
Final keyword.....	74
PHP Abstract class.....	75
Interface.....	76
MVC .....	77
Cake PHP.....	83

---

Learner's Sample Assignment .....	85
Sample Mini Project Title .....	88

## Overview of PHP

**PHP** is an acronym for "**PHP: Hypertext Preprocessor**". PHP is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. PHP is an interpreted language because there is no need to compile. It is a server-side scripting language its scripts are executed on the server. PHP is an object-oriented language.

PHP was originally created by a developer named **Rasmus Lerdorf**.

## Advantage of PHP

- PHP is platform independent software because it runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- It supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.).
- It is open source software. It is free to download and use.
- PHP is easy to learn and runs efficiently on the server side.

## PHP Syntax

- A PHP script starts with `<?php` and ends with `?>` and can be placed anywhere in the document.
- We can write PHP code within HTML code.

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

- We can also write HTML code within PHP code.

```
<?php
    echo "<h1>Hello World</h1>";
?>
```

**N.B:** A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

## Steps to run PHP

- Write any PHP code and save it by extension **.php** at root of wamp server in C:\Wamp\www\
- Write at the address bar of any browser the following path http:\\ localhost : <port number>

## PHP Printing

PHP **echo** and **print** both are used to display the output in PHP.

**echo:** The echo statement can be used with or without parentheses.

```
<?php
```

```
    echo "Hello Phoenix";  
    echo("Hello TCS iON");  
    ECHO "Hell Students";  
    EcHo "Sample Test";
```

```
?>
```

Basically **echo** is a language constructs not a function. If you want to use more than one parameter, it is required to use parenthesis.

Here **echo/ECHO/EcHo** statements are not case sensitive (all are legal).

**print :** The print statement can be used with or without parentheses.

```
<?php
```

```
    print "Hello Phoenix";  
    print("Hello TCS iON");  
    print("<h1>Hello Students</h1>");
```

```
?>
```

### Difference between echo and print in PHP:

#### echo:

- echo is a statement i.e. used to display the output. It can be used with parentheses echo or without parentheses echo.
- echo can pass multiple string separated as ( , )
- echo doesn't return any value.
- echo is faster than print.

#### print:

- print is also a statement i.e. used to display the output. It can be used with parentheses print( ) or without parentheses print.

- using print can doesn't pass multiple argument
- print always return 1.
- It is slower than echo.

## PHP Variable

A variable in PHP is a container that holds the data. Variable is just the identifier of the memory location.

### Rules for Variable declaration

- Variable in PHP starts with \$ sign followed by the name of the variable. (like \$name, \$x, \$f\_name).
- The variable name must begin with a letter or the underscore character.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_).
- A variable name must not be a predefine function and keywords
- A variable name should not contain space.
- Variable names are case-sensitive (\$friend and \$FRIEND are two different variables)

### Example of variable

```
<?php
$friend="Sumit";
echo "Hello:" .$friend. "<br>";
echo "Hello:" .$FRIEND. "<br>";
echo "Hello:" .$fRienD . "<br>";
?>
```

## PHP Concatenation

```
<?php
$friend="Sumit" ;
echo "Hello:" .$friend. "<br>" ;
?>
```

To concatenate string("Hello") with \$friend use **dot(.)** between string and variable name.

**Output:** Hello Sumit

### Inspecting Variable Contents:

PHP offers the **var\_dump( )** function, which accepts a variable and X-rays it for you.

## Constant in PHP

Constants are PHP container that remains unchanged or fixed during the execution of the script.



Constants are defined using PHP's define ( ) function

```
<?php
define("PI",3.14159);
print PI;
#Alternative way
define("pi",3.14159,true);
print pi;
?>
```

**N.B:**

**pi=>**specifies the name of the constant.

**3.14159=>**Specifies the value of the constant.

**true=>** Specifies whether the constant name should be case-insensitive. Default is false.

Constant name must follow the same rules as variable names, with one exception the "\$" prefix is not required for constant names.

## PHP Data Types

PHP is a Loosely Typed Language because PHP automatically converts the variable to the actual data type, depending on its value.

**gettype()** function is used to check only data type.

**var\_dump()** function is used to check value, data type and size use .

**There are 3 types of Data Type:**

- **Scalar(predefined): (It holds only single value)**
  - Integer
  - Float/double
  - String
  - Boolean
- **Compound(user-defined): (Multiple values in single variable)**
  - Array
  - Object
- **Special Data Type**
  - Null
  - Resource

**Examining the variable Type:**

- **boolean is\_int(mixed variable):** Check Whether it is integer or not. If the variable value is integer then it returns 1 otherwise none.
- **boolean is\_float(mixed variable):** Check Whether it is floating point or not. If the variable value is floating point then it returns 1 otherwise none.

- **boolean is\_bool(mixed variable):** Check Whether it is boolean or not. If the variable value is Boolean then it returns 1 otherwise none.
- **boolean is\_string(mixed variable):** Check Whether it is string or not. If the variable value is boolean then it returns 1 otherwise none.
- **boolean is\_array(mixed variable):** Check Whether it is array or not. If it is array then returns 1 otherwise none.
- **boolean is\_object(mixed variable):** Check Whether it is object type or not. If it is object then it returns 1 otherwise none.

### Example of Data types

```
<html>
<body>
<?php
$a=10;
$b=12.5;
$c="Hello";
$d='T';
$e=NULL;
$f=true;
echo $a."<br>";
echo $b."<br>";
echo $c."<br>";
echo $d."<br>";
echo $e."<br>";
echo $f."<br>";
echo gettype($a)."<br>";
echo gettype($b)."<br>";
echo gettype($c)."<br>";
echo gettype($d)."<br>";
echo gettype($e)."<br>";
echo gettype($f)."<br>";
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
var_dump($f);
echo "Integer:".is_int($a)."<br>";
echo "Float:".is_float($b)."<br>";
echo "String:".is_string($c)."<br>";
echo "Boolean:".is_bool($f)."<br>";
?>
</body>
</html>
```

**OUTPUT:**

```

10
12.5
Hello
T

1
integer
double
string
string
NULL
boolean
int 10
float 12.5
string 'Hello' (length=5)
string 'T' (length=1)
null
boolean true
Integer:1
Float:1
String:1
Boolean:1

```

**PHP Comments**

A comment is non-executable lines. Comment is used to write description of the program. Browser ignores the comment portion of the script.

**Type of Comments:**

1. Single line comments
2. Multiline Comment

**Example of PHP Comments**

```

<html>
<body>
<?php
    //This is a single line comment
    #This is also a single line comment
    /*
    It is used for multi line comment
    */
?>
</body>
</html>

```

**if else Statement**

if-else statement is used if we want to execute some statements when the condition is true and execute some other statements if the condition is false.

**Example of if else Statement**

```

<?php
$age=25;
if($age>=18)

```

```
{  
    echo "Eligible for vote";  
}  
else  
{  
    echo "Not Eligible for vote";  
}  
?>
```

## if elseif Statement

If elseif statement is used when program requires multiple testing conditions.

### Example of if elseif Statement

```
<?php  
$m=30;  
if($m>=80 && $m<=100)  
{  
    echo "Honours" ;  
}  
elseif($m>59)  
{  
    echo "First Div" ;  
}  
elseif($m>49)  
{  
    echo "Second Div" ;  
}  
elseif($m>=40)  
{  
    echo "Third Div" ;  
}  
else  
{  
    echo "Fail" ;  
}  
?>
```

## switch Statement

switch case is a multi-way branching system. In the program, if there is a possibility to make a choice from multiple options then switch statement is used.

The switch statement is useful for writing menu driven program.

### Example of switch Statement

```
<html>
<body>
<?php
$x=2;
switch ($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>
```

## Loop

A loop is defined as a block of statements which are repeatedly executed for certain number of times.

### for Loop

PHP for loops execute set of code for the specified number of times. When you know in advance how many times the script should run then it should be used.

#### Example of for Loop

```
<?php
for($i=0;$i<5;$i++)
{
    echo "Value of i=".$i."<br>";
}
?>
```

### while Loop

PHP while loops execute set of code while the specified condition is true.

### Example of while Loop

```
<?php
    $i=0;
    while($i<=10)
    {
        echo "this is line no ".$i."<br>";
        $i++;
    }
?>
```

### do while Loop

It executes the set of code at least once because condition is checked after executing the code and repeats the loop while the specified condition is true.

#### Example of do while Loop

```
<?php
    $i = 0;
    do
    {
        echo "Value of i=".$i;
        $i++;
    }while($i<=5);
?>
```

### foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array. For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

#### Example of foreach Loop:

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

#### OUTPUT:

```
one
two
three
```

## break Statement

PHP break statement terminate the execution of current for, while, do-while, switch and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.

### Example of break Statement

```
<?php
$val=1;
while ( $val<=5){
    if ($val ==3) {
        break;
    }
    echo "$val<br>";
}
?>
```

### OUTPUT:

1  
2

## continue Statement

Continue is used within looping structures to skip the rest of the current loop iteration" and continue execution at the condition evaluation and proceed to the beginning of the next iteration.

### Example of continue statement

```
<?php
$val=1;
while ( $val<=5){
    if (!($val % 2)) {
        continue;
    }
    echo $val;
}
?>
```

### OUTPUT: 1 3 5

## Array

PHP array is a special variable, which can hold more than one value at a time. It contains the value on the basis of index/key.

### Create an Array in PHP:

An array can be created by the array() language construct the index value can be assigned manually:

```
<?php
    $col = array();
    $col[0]="Green";
    $col[1]="Blue";
    $col[2]="Green";
    echo "Colours: " . $col[0] . ", " . $col[1] . " and " . $col[2] . ".";
?>
```

In PHP, there are three types of arrays:

- **Indexed Array:** Arrays with a numeric index which starts from 0. In this array values are stored and accessed in linear index order.

There are two ways to create indexed arrays:

```
<?php
    $col = array("Green", "Blue", "Red");
    echo "Colours: " . $col[0] . ", " . $col[1] . " and " . $col[2] . ".";
?>
```

- **Associative Array :** An array with strings as index. This stores element values in association with key values rather than index manner.

There are two ways to create an associative array:

```
$col = array("x"=>"Green", "y"=>"Blue", "Joe"=>"Red");
```

Or

```
$col['x'] = "Green";
$col['y'] = "Blue";
$col['z'] = "Red";
```

### Example of Associative Array

```
<?php
    $col = array("x"=>"Green", "y"=>"Blue", "Joe"=>"Red");
```



```

    echo "First Colour is: " . $col['x'];
?>

```

Print array element of Associative array using For each Loop:

```

<?php
    $col = array("x"=>"Green", "y"=>"Blue", "z"=>"Green");
    foreach($col as $p => $p_value) {
        echo "Key=" . $p . ", Value=" . $p_value;
        echo "<br>";
    }
?>

```

- **Multidimensional Array:** An array containing one or more 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 of Multidimensional Array

```

<?php
$model=array(
    array('Mi 4i',13000,20),

    array('Samsung',15000,10),
    array('Nokia',10000,15)
);
$c=count($model); //Count No. of rows

for($i=0;$i<$c;$i++)
{
    for($j=0;$j<count($model[$i]);$j++)
    {
        echo $model[$i][$j]. " ";
    }
    echo "<br>";
}
?>

```

### OUTPUT:

```

Mi 4i 13000 20
Samsung 15000 10
Nokia 10000 15

```

## Array Handling Function

- **count()**= How many number of element are present in the array.
- **array\_reverse()**= reverse of the array elements.
- **sort()**= sort the array elements in the ascending order.
- **rsort()**= sorting in descending order.
- **asort()**=sort associative arrays in ascending order, according to the value
- **ksort()**=sort associative arrays in ascending order, according to the key.
- **krsort ()**=sorting in key wise but in reverse order.
- **array\_search()**= searches the array for a given value and returns the corresponding key if successful.

### Example of Array Handling Function

```
<?php
$arr=array('Green','Violet','Blue','Orange','Black','Red');
//sort($arr);
rsort($arr);

for($i=0;$i<count($arr);$i++)
{
    echo $arr[$i];
    echo "<br>";
}
$day = array("Sun"=>"0","Fri"=>"6","Mon"=>"1","Tue"=>3 );
//asort($day);
//ksort($day);
//arsort($day);
krsort($day);
foreach($day as $x => $x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

## Function

PHP function is a block of code that can be reused multiple times in a program when required. A function will be executed by a call to the function. It can take input as argument list and return value.

While creating a function its name should starts with keyword “function” and the PHP code should be put inside “{}” braces.

**Types of Function:**

- 1. User Defined Function:** We can create our own functions.
- 2. Built-in function:** PHP has more than 1000 built-in functions.

**Create User defined Function:****Syntax:**

```
function Function_Name()
{
    //Body;
}
```

**Function without arguments:** A function will be executed by a call to the function without arguments.

```
<?php
function writeName()
{
    echo "Arup Karmakar";
}
writeName();
?>
```

**OUTPUT: Arup Karmakar**

**Function with arguments:** Data can be passed to functions through arguments. Arguments are specified inside the parentheses and we can add multiple arguments separated by commas.

```
<?php
function sum($x,$y)
{
    $z=$x+$y;
    echo "Result is: ".$z;
}
sum(10,20);
?>
```

**OUTPUT: Result is:30**

**PHP Default Argument Value:** If we call the function without arguments it takes the default value as argument.

```
<?php
function sum($x,$y=50)
{
    $z=$x+$y;
    echo "Result is: ".$z;
```

```

}
sum(10);
?>

```

**OUTPUT: Result is: 60**

**Function returning value:** PHP return statement immediately terminates the execution of a function when it is called from within that function and returns the value as well as flow to that point from where it is called.

```

<?php
function sum($x,$y)
{
    $z=$x+$y;
    return $z;
}
$r=sum(10,20);
echo "Result is: ".$r;
?>

```

**Output: Result is: 30**

**Local Scope of Function:** Scope of the local variable is only within the function in which it is declared.

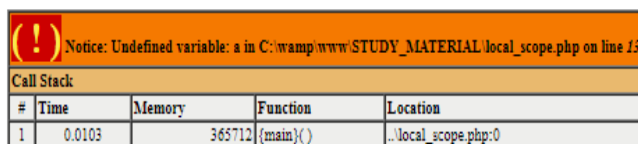
```

<?php
function m1()
{
    $a=10; // local scope
    echo "Inside the function:$a<br>";
}
m1();
echo "Outside the function:$a<br>";
?>

```

**OUTPUT:**

Inside the function:10



#	Time	Memory	Function	Location
1	0.0103	365712	{main}()	..local_scope.php:0

Outside the function:

**Another Example:**

```

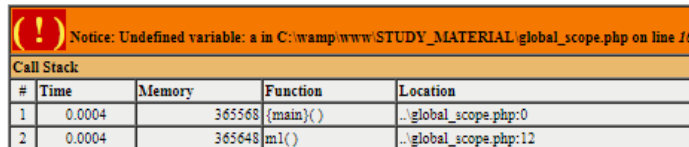
<?php
$a=10;
function m1() {
    echo "Inside the function: ".$a."<br>";
}

```

```

}
m1();
    echo "Outside the function: ".$a."<br>";
?>

```

**OUTPUT:**


Notice: Undefined variable: a in C:\wamp\www\STUDY\_MATERIAL\global\_scope.php on line 10

#	Time	Memory	Function	Location
1	0.0004	365568	{main}()	..\global_scope.php:0
2	0.0004	365648	m1()	..\global_scope.php:12

Inside the function:  
 Outside the function:10

**Global Scope:** Variable declared outside the function or module is known as global variable. The global keyword is used before the variables to access a global variable from within a function.

```

<?php
$a = 10;
$b = 20;

function m1() {
    global $a, $b;
    $s = $a + $b;
}
m1();
echo $s; // outputs 30
?>

```

**OUTPUT:** 30

**Static Scope:** Each time the function is called; static variable contains the information of previous function call.

```

<?php
function m1()
{
    static $c = 1;
    echo $c. "<br>";
    $c++;
}
m1();
m1();
m1();
?>

```

**OUTPUT:**

```

1
2
3

```

## General Function

**include():** include() function is used to includes a file in the current php file. If an error occurs, the include() function generates a warning, but the script will continue execution.

```
<?php
    include("header.php");
    echo "<h1>Welcome to my home page!</h1>";
?>
```

**require():** The require() function is identical to include(), except that it handles errors differently.

```
<?php
    require("header.php");
    echo "<h1>Welcome to my home page!</h1>";
?>
```

### Difference between include() and require():

require() and include() performing the same function, however the major difference is require() will throw a fatal error (E\_COMPILE\_ERROR ) since it is not able to find the specific file on specified location and it will stop the execution as a result, on the other hand include() will show the warning (E\_WARNING) however it will not stop the script execution.

**include\_once():** include\_once() include the file only once at a time.

### Functional Difference:

There is only one difference between these two functions. If the code a from a file has been already included then it will not be included again if we use include\_once(). Means include\_once() include the file only once at a time.

**isset():** It determines whether a variable has been declared and loaded with a value by a programmer, returns true if such a variable is passed to it, if not then it returns false.

```
<?php
    $a="Hello";
    var_dump(isset($a));
    unset($a);
    var_dump(isset($a));
?>
```

**OUTPUT:**


```
boolean true
boolean false
```

**unset():** The unset() function destroys a given variable.

```
<?php
$a="Hello";
echo $a;
unset($a);
echo $a;
?>
```

**OUTPUT:**

Hello

 Notice: Undefined variable: a in C:\wamp\www\STUDY_MATERIAL\unset.php on line 11				
Call Stack				
#	Time	Memory	Function	Location
1	0.0003	364312	{main}()	..\unset.php:0

**empty():** The empty() function is used to check whether a variable is empty or not.

```
$x=10;
$y=20;
if(empty($x) || empty($y))
{
    echo "Empty field";
}
else
{
    $z=$x+$y;
}
```

**header():** The header() function sends a raw HTTP header to a client.

File Redirection in PHP using header()

```
<?php
header( 'Location: http://www.yoursite.com/new_page.html' ) ;
?>
```

Change the code on the redirect page to be simply this. You need to replace the URL above With the URL you wish to direct to.

#### Auto Redirect:

```
<?php
header("Refresh:5;URL=dc.php");//this will redirect to dc.php
automatically after 5 sec...
echo "After 5 sec you will be no longer exist in this page";
?>
```

### String Function

- **crypt():** It is used to create hashing string One-way.
- **md5():** It is used to calculate the MD5 hash of a string.
- **ord():** It is used to return ASCII value of the first character of a string.
- **explode():** Break a string into an array
- **implode() :** The implode() function returns a string from elements of an array.
- **strtolower():** The strtolower() function returns string in lowercase letter.
- **strtoupper():** The strtoupper() function returns string in uppercase letter.
- **ucfirst():** The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.
- **strrev():** The strrev() function returns reversed string.
- **strlen():** The strlen() function returns length of the string.
- **trim():** It is used to remove whitespace.
- **ltrim():** It is used to remove whitespace from the left side of a string.
- **rtrim():** It is used to remove whitespace from the right side of a string.
- **str\_pad():** Pads a string to a new length
- **strpos():** Returns the position of the first occurrence of a string inside another string (case-sensitive).
- **strcmp():** Compares two strings (case-sensitive).

#### Example of String Function

```
<?php
echo "Crypt Example:".crypt(123)."<br>";
echo "md5 Example:".md5("Hello")."<br>";
echo "ord Example:".ord('A')."<br>";
echo "Lower Case:".strtolower('HELLO')."<br>";
echo "Upper Case:".strtoupper('hello')."<br>";
$str=" Hello World ";
echo "Full String length:".strlen($str)."<br>";
echo "With ltrim:".strlen(ltrim($str))."<br>";
echo "With rtrim:".strlen(rtrim($str))."<br>";
echo "With trim:".strlen(trim($str))."<br>";
$str = "Hello World";
echo str_pad($str,20,"*")."<br>";
echo str_pad($str,20,"*",STR_PAD_LEFT)."<br>";
```



```

echo str_pad($str,20,"*",STR_PAD_BOTH)."<br>";
echo "Reverse String:".strrev($str)."<br>";
echo strcmp("Hello","Hello")."<br>";//Output:0[=0,if the two strings are
equal]
echo strcmp("Hello123","Hello")."<br>"; //Output:3 [>0 if string1 is greater
than string2]
echo strcmp("Hello","Hello123")."<br>"; //Output:-3 [<0 if string1 is less
than string2]
echo strcmp("Hello","abc")."<br>"; //output:-1
echo strcmp("abc","Hello")."<br>"; //output:1
echo strpos("Hello world!","o")."<br>"; //Index position
echo strtolower("Hello WORLD")."<br>";
echo strtoupper("Hello WORLD!")."<br>";
?>

```

### OUTPUT:

```

Crypt Example:$1$AE3.J30.$KQ.GS3svotxgeRfsOSMuQ/
md5 Example:8b1a9953c4611296a827abf8c47804d7
ord Example:65
Lower Case:hello
Upper Case:HELLO
Full String length:13
With ltrim:12
With rtrim:12
With trim:11
Hello World*****
*****Hello World
****Hello World****
Reverse String:dlroW olleH
0
3
-3
-1
1
4
hello world.
HELLO WORLD!

```

### Example of explode() function

```

<?php
//Break a string into an array
$str = "Hi Mr. Roy. How are you.";

print_r (explode(" ", $str));
print "<br>";
print_r (explode(".", $str))."<br>";
print "<br>";
$str = 'Green,Red,Blue,Yellow';
// zero limit
print_r(explode(',', $str, 0));

```

```

print "<br>";
// positive limit
print_r(explode(' ', $str, 3));
print "<br>";
print_r(explode(' ', $str, 5));
print "<br>";
// negative limit
print_r(explode(' ', $str, -2));
print "<br>";
print_r(explode(' ', $str, -3));
print "<br>";
?>

```

### OUTPUT:

```

Array ( [0] => Hi [1] => Mr. [2] => Roy. [3] => How [4] => are [5] => you. )
Array ( [0] => Hi Mr [1] => Roy [2] => How are you [3] => )
Array ( [0] => Green,Red,Blue, Yellow )
Array ( [0] => Green [1] => Red [2] => Blue, Yellow )
Array ( [0] => Green [1] => Red [2] => Blue [3] => Yellow )
Array ( [0] => Green [1] => Red )
Array ( [0] => Green )

```

### Example of implode() function

```

<?php
$arr = array('Red', 'Green', 'Blue', 'Orange');
echo implode(" ", $arr). "<br>";
echo implode("+", $arr). "<br>";
echo implode("-", $arr). "<br>";
echo implode("X", $arr);
?>

```

### OUTPUT:

```

Red Green Blue Orange
Red+Green+Blue+Orange
Red-Green-Blue-Orange
RedXGreenXBlueXOrange

```

## Date Function

Date function is used to print various date formats.

### Example of Date Function

```

<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";

```

```
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l")."<br>"; //display todays name
echo "Leap year or not: " . date("L")."<br>"; //check leap year or not
echo "Day of the month:" . date("j")."<br>"; //display the day of the month
echo "Year:".date("Y")."<br>"; //Year (four digits)
echo "Year:".date("y")."<br>"; //Year (two digits)
echo "Month:".date("M")."<br>"; //Month of year (three letters)
echo "Month:".date("m")."<br>"; //Month of year (number - leading zeroes)
echo "Month Name:".date("F")."<br>"; //Month name(Full)
echo "Day:".date("D")."<br>"; //Day of week (three letters)
echo "Day:".date("d")."<br>"; //Day of month, a number with leading zeroes
echo "Day of the Year:".date("Z")."<br>"; //Day of year
echo "Day offset in seconds:".date("z")."<br>"; //Offset in seconds from GMT
echo "The time is: " . date("h:i:sa")."<br>"; //'am' or 'pm' lowercase
echo "The time is: " . date("H:i:SA")."<br>"; //Hour (24-hour format - leading zeroes)
echo "The time is: " . date("h:i:SA")."<br>"; //'am' or 'pm' upercase
?>
```

### OUTPUT:

```
Today is 2018/09/05
Today is 2018.09.05
Today is 2018-09-05
Today is Wednesday
Leap year or not: 0
Day of the month:5
Year:2018
Year:18
Month:Sep
Month:09
Month Name:September
Day:Wed
Day:05
Day of the Year:0
Day offset in seconds:247
The time is: 06:17:21am
The time is: 06:17:21AM
The time is: 06:17:21AM
```

## Math Function

- **abs():** Returns the absolute (positive) value of a number.
- **ceil():** Rounds a number up to the nearest integer.
- **floor():** Rounds a number down to the nearest integer.
- **pow():** Returns x raised to the power of y.

- **rand():** Generates a random integer.
- **round():** Rounds a floating-point number.
- **sqrt():** Returns the square root of a number.

### Example of Math Function

```
<?php
```

```
echo "Absolute Value:".abs(-5)."<br>";  
echo "Ceil Value:".ceil(123.58)."<br>";  
echo "Floor Value:".floor(123.58)."<br>";  
echo "Pow Value:".pow(2,5)."<br>";  
echo "Rand Value:".rand(1,5)."<br>";  
echo "Round Value:".round(123.58)."<br>";  
echo "Square root Value:".sqrt(64)."<br>";
```

```
?>
```

### OUTPUT:

```
Absolute Value:5  
Ceil Value:124  
Floor Value:123  
Pow Value:32  
Rand Value:3  
Round Value:124  
Square root Value:8
```

## PHP Super globals

PHP super global variable is used to access global variables from anywhere in the PHP script. PHP Super global variable is accessible inside the same page that defines it, as well as outside the page.

These super global variables are:

**\$GLOBALS:** \$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script.

PHP stores all global variables in an array called \$GLOBALS[index]. The index holds the name of the variable.

### Example of \$GLOBALS

```
<?php
```

```
$a=10;  
$b=20;  
function sum() {  
    $GLOBALS['c'] = $GLOBALS['a'] + $GLOBALS['b'];  
}  
sum();  
echo $c;  
?>
```

**OUTPUT: 30**

Here c is a variable within the **\$GLOBALS** array. Variable can be easily accessible from outside the function!

**\$\_SERVER:** Server and execution environment information.

**Example of \$\_SERVER**

```
<?php
echo $_SERVER[ 'PHP_SELF' ];
echo "<br>";
echo $_SERVER[ 'SERVER_NAME' ];
echo "<br>";
echo $_SERVER[ 'HTTP_HOST' ];
echo "<br>";
echo $_SERVER[ 'HTTP_REFERER' ];
echo "<br>";
echo $_SERVER[ 'SERVER_PORT' ];
echo "<br>";
echo $_SERVER[ 'SCRIPT_NAME' ];
?>
```

**OUTPUT:**

```
/TEST/SERVERphp.php
localhost
localhost
http://localhost/TEST/
80
/TEST/SERVERphp.php
```

**Details:**

**\$\_SERVER['PHP\_SELF']** - Returns the filename of the currently executing script.

**\$\_SERVER['SERVER\_NAME']** - Returns the name of the host server.

**\$\_SERVER['HTTP\_HOST']** - Returns the Host header from the current request

**\$\_SERVER['HTTP\_REFERER']** - Returns the complete URL of the current page (not reliable because not all user-agents support it).

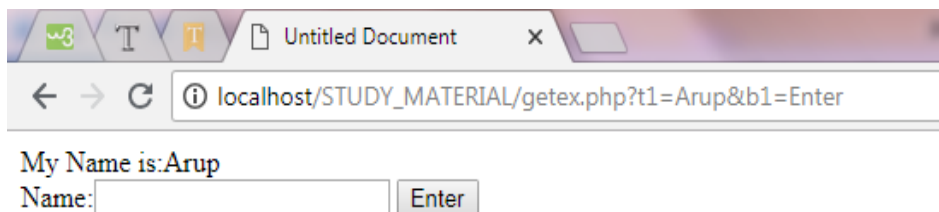
**\$\_SERVER['SCRIPT\_NAME']** - Returns the path of the current script

**\$\_SERVER['SERVER\_PORT']** - Returns the port on the server machine being used by the web server for communication.

**\$\_REQUEST** : An associative array that by default contains the contents of **\$\_GET**, **\$\_POST** and **\$\_COOKIE**.

**Example of \$\_REQUEST**

```
<html>
<?php
if(isset($_REQUEST['b1']))
{
    $nm=$_REQUEST['t1'];
    echo "My Name is: ".$nm;
}
?>
<body>
<form name="f1" action="">
Name:<input type="text" name="t1">
<input type="submit" value="Enter" name="b1">
</form>
</body>
</html>
```

**OUTPUT:**

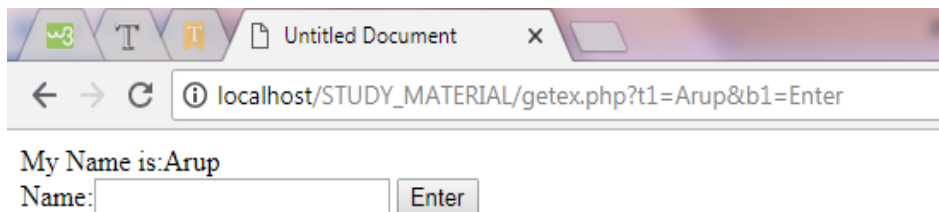
**\$\_GET["FormElementName"]** : It is used to receive value from a HTML form(script) via 'get' method. Data sent from a form via get method is visible to everyone (it display on the browser URL bar).

**Example of \$\_GET**

```
<html>
<?php
if(isset($_GET['b1']))
{
    $nm=$_GET['t1'];
    echo "My Name is: ".$nm;
}
?>
<body>
<form name="f1" method="get" action="">
Name:<input type="text" name="t1">
```

```
<input type="submit" value="Enter" name="b1">
</form>
</body>
</html>
```

### OUTPUT:



**\$\_POST["FormElementName"]:** It is used to receive value from a HTML form(script) via 'post' method. Data sent from a form via post method is invisible to others ( Data is invisible on the browser url bar ).

### Example of \$\_POST

```
<html>
<?php
    if(isset($_POST['b1']))
    {
        $x=$_POST['t1'];
        $y=$_POST['t2'];
        if(empty($x) || empty($y))
        {
            echo "Empty field";
        }
        else
        {
            $z=$x+$y;
            echo "Result is: ".$z;
        }
    }
?>
<body>
<form name="f1" method="post" action="">
<pre>
Enter Number 1:<input type="text" name="t1">
Enter Number 2:<input type="text" name="t2">
<input type="submit" name="b1" value="Submit">
</form></body></html>
```

**OUTPUT:**

Result is:30

Enter Number 1:	<input type="text" value="10"/>
Enter Number 2:	<input type="text" value="20"/>
<input type="button" value="Submit"/>	

- **\$\_FILES:** An associative array of items uploaded to the current script via the HTTP POST method.
- **\$\_COOKIE:** An associative array of variables passed to the current script via HTTP Cookies.
- **\$\_SESSION:** An associative array containing session variables available to the current script.
- **\$\_ENV:** An associative array of variables passed to the current script via the environment method.

**Cookie**

A cookie is often used to identify a user. Cookies are text files stored on the client computer and they are kept of use tracking purpose. Each time the same computer requests a page with a browser, it will send the cookie too.

There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

**Create a Cookie:****Syntax:**

```
setcookie(name, value, expire, path, domain, security);  
<?php  
    setcookie("user", "Phoenix Enterprise", time()+3600, "/", "", 0);  
?>
```

Here is the detail of all the arguments:

**Name** – This sets the name of the cookie and is stored in an environment variable called HTTP\_COOKIE\_VARS. This variable is used while accessing cookies.

**Value** – This sets the value of the named variable and is the content that you actually want to store.



**Expiry** – This specifies a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.

**Path** – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

**Domain** – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.

**Security** – 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 means cookie can be sent by regular HTTP.

### Retrieve a Cookie

```
<?php
// Print a cookie
echo $_COOKIE["user"];
// A way to view all cookies
print_r($_COOKIE);
?>
```

### Delete a Cookie

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

### Example of Cookie

#### takecookie.php:

```
<form name="form1" method="post" action="writecookie.php">
SETTING COOKIES VARIABLE
<pre>
USER ID<br/>
<input type="text" name="t1"/>
PASSWORD<br/>
<input type="password" name="t2"/>
<input type="submit" value="send"/>
</pre>
</form>
```

#### writecookie.php

```
<?php
$user_id = $_POST['t1'];
$pass = $_POST['t2'];
```

```
        setcookie("id",$user_id,time()+3600,"/","",0);
        setcookie("pass",$pass,time()+3600,"/","",0);
?>
<a href="readcookie.php">Read Cookies information ....</a>
```

**readcookie.php:**

```
<?php
    if(isset($_COOKIE["id"]) && isset($_COOKIE["pass"]))
    {
        echo "Welcome ".$_COOKIE["id"]."identified as
".$_COOKIE["pass"]."<br/>";
    }
    else
    {
        echo "Not a Authenticated user"."<br/>";
    }
?>
<a href="destroycookie.php" >Delete cookies .....</a>
```

**destroycookie.php:**

```
<?php
    setcookie("id","",time()-60,"/","",0);
    setcookie("pass","",time()-60,"/","",0);
    echo "cookies deleted from computer..."
?>
```

**OUTPUT:**

SETTING COOKIES VARIABLE

USER ID

CMC

PASSWORD

...

send

[Read Cookies information ....](#)

Welcome CMCidentified as 123

[Delete cookies .....](#)

cookies deleted from computer...

**Session**

To make data accessible across the various pages of an entire website we use PHP session. A session creates a file in temporary directory on server to store session variables, this data stored in session variable will be available to all pages on the site during that visit.

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 3c7foj34fgh65op658ssddddf8547w33.
- 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\_id 5c7foj34554h65op658ssddddf8547w33.

When a php script wants to retrieve the value from a session variable, php automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A php session is easily started by making a call to the session\_start() function. These function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session\_start () at the beginning of the page.

Session variables are stored in associative array called \$\_SESSION[].These variables can be accessed during lifetime of a session.

A php session can be destroyed by session\_destroy() function. This function does not need any argument and single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset() function to unset a session variable.

### Create Session:

```
<?php
session_start();
$_SESSION['user']=$uid;
?>
```

### Fetch Session Value:

```
<?php
session_start();
$user_name=$_SESSION['user'];
?>
```

### Destroy Session:

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

### Example of Session

#### login.php:

```
<html>
<body>
<form name="f1" method="post" action="home_code.php">
User's Login<br/>
<pre>

User ID :<input type = "text" name="t1" />
Password:<input type ="password" name="t2" />
</pre>
<input type="submit" value ="Login" />
</form></body>
</html>
```

User's Login

User ID :	Phoenix
Password:	***

**home\_code.php:**

```
<?php
session_start();
$uid = $_POST['t1'];
$pas = $_POST['t2'];
if($uid == "Phoenix" && $pas == "123")
{
    if(isset($_SESSION['counter']))
    {
        $_SESSION['counter'] +=1;
    }
    else
    {
        $_SESSION['counter']=1;
    }
    $_SESSION['user']=$uid;
    header('Location:home.php');
}
else{
    echo "Invalid Userid or password";
}
?>
```

**home.php:**

```
<?php
session_start();

if($_SESSION['user']==NULL)
    header ('Location:login.php');
else
{
    $user_name=$_SESSION['user'];

    echo "Welcome to TCS iON , History : You have visited
    ".$_SESSION['counter']." times(s) in this session.<br>";
    echo "<br>Hi ".$user_name;
}
?>
<br>
```

```
<a href="logout.php">LOG OUT</a>
```

### logout.php:

```
<?php
session_start();
session_unset();
session_destroy();
echo "You have successfully logged out";
?>
<a href="second.php">LOGIN AGAIN</a>
```

---

Welcome to TCS iON , History : You have visited 6 times(s) in this session.

Hi Phoenix  
[LOG OUT](#)

## File Handling

File handling is an important part of any web application. PHP File System allows us to perform different task like create file, read file line by line, read file character by character, write file, append file, delete file and close file etc.

### File Handling Function

- **fopen():** fopen() function is used to open files.
- **fgets():** fgets() function is used to read a single line from a file.
- **fread():** The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.
- **fwrite():** The PHP fwrite() function is used to write content of the string into file.
- **fclose():** fclose() function is used to close an open file.
- **feof():** feof() function checks if the "end-of-file" (EOF) has been reached.
- **unlink():** The PHP unlink() function is used to delete file.

### File Opening Mode

**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:** Append-Opens and writes to the end of the file or creates a new file if it doesn't exist.

**a+:** Read/Append-Preserves file content by writing to the end of the file.

**x:** Write only-Creates a new file. Returns FALSE and an error if file already exists.

**x+:** Read/Write-Creates a new file. Returns FALSE and an error if file already exists.

### Example of Reading a File Line by Line

```
<?php
$file = fopen("a.txt", "r") or exit("Unable to open file!");
while(!feof($file))
{
    echo fgets($file). "<br />";
}
fclose($file);
?>
```

### Example of Checking Directory

```
<?php
$d_name = "Session";//directory name
if(file_exists($d_name))
{
    $f_size = filesize($d_name);
    $msg = "Directory exist with name $d_name";
    echo ($msg);
}
else
{
    echo ("file $d_name doesn't exist");
}
?>
```

### Example of Create a new file append text in it

```
<?php
$f_name = "textfile.txt";
$fpt = fopen($f_name,"w");
if($fpt == false)
{
    echo "Error in opening new file";
    exit();
}
fwrite($fpt,"This is a simple Text File");
fclose($fpt);
if(file_exists($f_name))//file_exist() function confirms existence of a
file/directory..
{
    $f_size = filesize($f_name);
    $msg = "File Created successfully with name $f_name ";
}
```

```

    $msg .= "Containing $f_size bytes";
    echo ($msg);
}
else
{
    echo "FILE $f_name does not exist. ";
}
?>

```

### Example of File Upload

#### upload.html:

```

<body>
SELECT A FILE TO UPLOAD <br>
<form      name="form1"      method="post"      action="file_upload.php"
enctype="multipart/form-data">
<input type="file" name="file" size="50"/><br/>
<input type="submit" value="Upload File"/>
</form>
</body>

```

#### File\_upload.php:

```

<?php
$target ="c:\wamp\www\up\\";//up folder created in server (under www) to
store upload file
$target = $target.basename($_FILES['file']['name']);
echo "File  uploading".$target."<br>";
echo "Temporary File Created by php:".$_FILES['file']['tmp_name']."<br/>";
if(move_uploaded_file($_FILES['file']['tmp_name'],$target)){
    echo "The File ".basename($_FILES['file']['name']). "Has been Created";
}
else{
    echo "Sorry , there was a problem uploading your file";
}
?>
<html><head><title>UPLOADING COMPLETED</title></head>
<body><h2>UPLOADING FILE INFO</h2>
<ul><li>SENT FILE :<?php echo $_FILES['file']['name']; ?> </li>
<li>FILE SIZE :<?php echo $_FILES['file']['size']; ?> </li>
<li>FILE TYPE :<?php echo $_FILES['file']['type']; ?></li></ul>
</body></html>

```



**OUTPUT:**

```
File uploadingc:\wamp\www\up\2OpenDOSShell.png
Temporary File Created by php:C:\wamp\tmp\php5F75.tmp
The File 2OpenDOSShell.pngHas been Created
```

**UPLOADING FILE INFO**

- SENT FILE :2OpenDOSShell.png
- FILE SIZE :314393
- FILE TYPE :image/png

**Example of File download(Text File)****download1.php**

```
<?php
$file_url = 'http://www.phoenix.com/f.txt';
header('Content-Type: application/octet-stream');
header("Content-Transfer-Encoding: utf-8");
header("Content-
disposition: attachment; filename=\"\" . basename($file_url) . \"\");
readfile($file_url);
?>
```

**MySQL Database**

Database MySQL officially, but also commonly "My Sequel" is a relational Database Management System (RDBMS) that runs as a server providing multi user access to a number of databases. The SQL phrase stands for Structured Query Language. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used WAMP web application software stack—WAMP is an acronym for "Windows, Apache, MySQL, and PHP".

**Create User**

```
CREATE USER 'arin'@'localhost' IDENTIFIED BY '***';
```

**Grant Privileges**

```
GRANT ALL PRIVILEGES ON * . * TO 'arin'@'localhost' IDENTIFIED BY '***' WITH
GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0 ;
```

## Drop User

```
DROP USER 'arin'@'localhost';
```

## Create Database

```
CREATE DATABASE `new_db` ;
```

## Drop Database

```
DROP DATABASE `test`;
```

## Create Table

```
CREATE TABLE `new_db`.`student` (  
  `Roll` INT( 4 ) NOT NULL ,  
  `Name` VARCHAR( 30 ) NOT NULL ,  
  `Marks` FLOAT( 2, 2 ) NOT NULL  
  ) ENGINE = InnoDB;
```

## Set Primary Key

```
ALTER TABLE `student` ADD PRIMARY KEY ( `Roll` );
```

## Drop Table

```
DROP TABLE `a`
```

## Auto Increment

- ```
CREATE TABLE animals ( id MEDIUMINT NOT NULL AUTO_INCREMENT, name CHAR(30) NOT NULL, PRIMARY KEY (id) );
```
- ```
INSERT INTO animals (name) VALUES ('dog'),('cat'),('penguin'), ('lax'),('whale'),('ostrich');
```
- ```
SELECT * FROM animals;
```

 Which returns:

## Insert some values to dept

```
INSERT INTO `new_db`.`dept` (`dept_no`, `d_name`, `d_loc`) VALUES ('10', 'IT', 'Kolkata'), ('20', 'MKT', 'Delhi'), ('30', 'E&T', 'Mumbai');
```

## Select Statement

The **SELECT** statement is used to pull information from a table. The general form of the statement is:

```
SELECT what_to_select
```

```
FROM which_table WHERE conditions_to_satisfy;
```

### Some Select Statements

- `mysql> SELECT * FROM pet;`
- `mysql> SELECT name, birth FROM pet;`
- `mysql> SELECT owner FROM pet;`
- `mysql> SELECT DISTINCT owner FROM pet;`
- `mysql> SELECT * FROM pet WHERE name = 'Bowser';`
- `mysql> SELECT * FROM pet WHERE birth >= '1998-1-1';`
- `mysql> SELECT * FROM pet WHERE species = 'dog' AND sex = 'f';`
- `mysql> SELECT * FROM pet WHERE species = 'snake' OR species = 'bird';`
- `mysql> SELECT * FROM pet WHERE (species = 'cat' AND sex = 'm') -> OR  
(species = 'dog' AND sex = 'f');`
- `mysql> SELECT name, species, birth FROM pet -> WHERE species = 'dog' OR  
species = 'cat';`

### Eliminating duplicate rows

Eliminate duplicate rows by using the distinct keyword in the select clause.

Select distinct dept\_no from emp;

### Limiting the rows selected

Restrict the rows returned by using the where clause.

The where clause follows the from clause.

`select * from emp where dept_no=20;`

## Comparison Operator

| Operator | Meaning                  |
|----------|--------------------------|
| =        | Equal to                 |
| >        | Greater than             |
| >=       | Greater than or equal to |
| <        | Less than                |
| <=       | Less than or equal to    |
| <>       | Not equal to             |

| Operator             | Meaning                         |
|----------------------|---------------------------------|
| BETWEEN<br>...AND... | Between two values (inclusive), |
| IN(set)              | Match any of a list of values   |
| LIKE                 | Match a character pattern       |
| IS NULL              | Is a null value                 |

### Query:

```
select * from emp where salary between 5000 and 10000;
```

## Pattern Matching

- mysql> SELECT \* FROM pet WHERE name LIKE 'b%';
- mysql> SELECT \* FROM pet WHERE name LIKE '%fy';
- mysql> SELECT \* FROM pet WHERE name LIKE '%w%';
- mysql> SELECT \* FROM pet WHERE name LIKE '\_\_\_\_';

## Logical Operator

| Operator | Meaning                                                   |
|----------|-----------------------------------------------------------|
| AND      | Returns TRUE if <i>both</i> component conditions are true |
| OR       | Returns TRUE if <i>either</i> component condition is true |
| NOT      | Returns TRUE if the following condition is false          |

### Query:

```
select * from emp where dept_no=10 or dept_no=20;
```

## Order by

Sort rows with the ORDER BY clause

ASC: Ascending order, default

DESC: Descending order

The order by clause comes last in the SELECT statement

### Query:

```
select * from emp order by salary;
```

### Sorting by Multiple Columns

The order of ORDER BY list is the order of sort.

You can sort by a column that is the SELECT list.

### Query:

```
select * from emp order by dept_no,salary;
```

## String Function

| Function                                                                 | Purpose                                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>LOWER(column/ expression)</code>                                   | Converts alpha character values to lowercase                                                                                                                                                                                                                                        |
| <code>UPPER(column/ expression)</code>                                   | Converts alpha character values to uppercase                                                                                                                                                                                                                                        |
| <code>INITCAP(column/ expression)</code>                                 | Converts alpha character values to uppercase for the first letter of each word, all other letters in lowercase                                                                                                                                                                      |
| <code>CONCAT(column1/ expression1<br/>,<br/>column2/ expression2)</code> | Concatenates the first character value to the second character value; equivalent to concatenation operator (  )                                                                                                                                                                     |
| <code>SUBSTR(column/ expression, m<br/>[, n])</code>                     | Returns specified characters from character value starting at character position <i>m</i> , <i>n</i> characters long (If <i>m</i> is negative, the count starts from the end of the character value. If <i>n</i> is omitted, all characters to the end of the string are returned.) |

| Function                                                                                                         | Purpose                                                                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>LENGTH(column/ expression)</code>                                                                          | Returns the number of characters in the expression                                                                                                                                                                                                                                          |
| <code>INSTR(column/ expression,<br/>'string', [,m], [n] )</code>                                                 | Returns the numeric position of a named string. Optionally, you can provide a position <i>m</i> to start searching, and the occurrence <i>n</i> of the string. <i>m</i> and <i>n</i> default to 1, meaning start the search at the beginning of the search and report the first occurrence. |
| <code>LPAD(column/ expression, n,<br/>'string')</code><br><code>RPAD(column/ expression, n,<br/>'string')</code> | <code>LPAD</code> Pads the character value right-justified to a total width of <i>n</i> character positions<br><code>RPAD</code> Pads the character value left-justified to a total width of <i>n</i> character positions                                                                   |
| <code>TRIM(leading/trailing/both<br/>, trim_character FROM<br/>trim_source)</code>                               | Enables you to trim heading or trailing characters (or both) from a character string. If <i>trim_character</i> or <i>trim_source</i> is a character literal, you must enclose it in single quotes.<br>This is a feature available from Oracle8i and later.                                  |
| <code>REPLACE(text,<br/>search_string,<br/>replacement_string)</code>                                            | Searches a text expression for a character string and, if found, replaces it with a specified replacement string                                                                                                                                                                            |

## Example of String Function

These functions manipulate character strings:

| Function                                 | Result     |
|------------------------------------------|------------|
| <code>CONCAT('Hello', 'World')</code>    | HelloWorld |
| <code>SUBSTR('HelloWorld', 1, 5)</code>  | Hello      |
| <code>LENGTH('HelloWorld')</code>        | 10         |
| <code>INSTR('HelloWorld', 'W')</code>    | 6          |
| <code>LPAD(salary, 10, '*')</code>       | *****24000 |
| <code>RPAD(salary, 10, '*')</code>       | 24000***** |
| <code>TRIM('H' FROM 'HelloWorld')</code> | elloWorld  |

## Number Function

| Function                                 | Purpose                                                                                                                                                                                         |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ROUND(column expression, n)</code> | Rounds the column, expression, or value to <i>n</i> decimal places, or, if <i>n</i> is omitted, no decimal places. (If <i>n</i> is negative, numbers to left of the decimal point are rounded.) |
| <code>TRUNC(column expression, n)</code> | Truncates the column, expression, or value to <i>n</i> decimal places, or, if <i>n</i> is omitted, then <i>n</i> defaults to zero                                                               |
| <code>MOD(m, n)</code>                   | Returns the remainder of <i>m</i> divided by <i>n</i>                                                                                                                                           |

## Date Function

| Function                    | Description                        |
|-----------------------------|------------------------------------|
| <code>MONTHS_BETWEEN</code> | Number of months between two dates |
| <code>ADD_MONTHS</code>     | Add calendar months to date        |
| <code>NEXT_DAY</code>       | Next day of the date specified     |
| <code>LAST_DAY</code>       | Last day of the month              |
| <code>ROUND</code>          | Round date                         |
| <code>TRUNC</code>          | Truncate date                      |

- MONTHS\_BETWEEN ('01-SEP-95','11-JAN-94')  
➔ 19.6774194
- ADD\_MONTHS ('11-JAN-94',6) ➔ '11-JUL-94'
- NEXT\_DAY ('01-SEP-95','FRIDAY')  
➔ '08-SEP-95'
- LAST\_DAY('01-FEB-95') ➔ '28-FEB-95'

## Data Type Conversion

| Function                                                                      | Purpose                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TO_CHAR( <i>number</i>   <i>date</i> , [ <i>fmt</i> ] , [ <i>nlsparams</i> ]) | <b>Date Conversion:</b> The <i>nlsparams</i> parameter specifies the language in which month and day names and abbreviations are returned. If this parameter is omitted, this function uses the default date languages for the session.                                               |
| TO_NUMBER( <i>char</i> , [ <i>fmt</i> ] , [ <i>nlsparams</i> ])               | Converts a character string containing digits to a number in the format specified by the optional format model <i>fmt</i> .<br><br>The <i>nlsparams</i> parameter has the same purpose in this function as in the TO_CHAR function for number conversion.                             |
| TO_DATE( <i>char</i> , [ <i>fmt</i> ] , [ <i>nlsparams</i> ])                 | Converts a character string representing a date to a date value according to the <i>fmt</i> specified. If <i>fmt</i> is omitted, the format is DD-MON-YY.<br><br>The <i>nlsparams</i> parameter has the same purpose in this function as in the TO_CHAR function for date conversion. |

## Date Formats

## Elements of the Date Format Model

|              |                                                         |
|--------------|---------------------------------------------------------|
| <b>YYYY</b>  | <b>Full year in numbers</b>                             |
| <b>YEAR</b>  | <b>Year spelled out</b>                                 |
| <b>MM</b>    | <b>Two-digit value for month</b>                        |
| <b>MONTH</b> | <b>Full name of the month</b>                           |
| <b>MON</b>   | <b>Three-letter abbreviation of the month</b>           |
| <b>DY</b>    | <b>Three-letter abbreviation of the day of the week</b> |
| <b>DAY</b>   | <b>Full name of the day of the week</b>                 |
| <b>DD</b>    | <b>Numeric day of the month</b>                         |



### TO\_CHAR Function with Dates

```
SELECT last name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
       AS HIREDATE  
FROM   employees;
```

## Aggregate Functions

**AVG() Function :** The AVG() function returns the average value of an expression.

```
select avg(salary) 'Average Salary' from emp;
```

**COUNT() Function:** The COUNT() function returns the count of rows in a table.

```
select count(emp_no) 'Total No Of Employee' from emp;
```

**MAX() Function:** The MAX() function returns the largest value of the selected column.

```
select max(salary) 'Maximum Salary' from emp;
```

**MIN() Function:** The MIN() function returns the smallest value of the selected column.

```
select min(salary) 'Minimum Salary' from emp;
```

**SUM() Function:** The SUM() function returns the total sum of a numeric column.

```
select sum(salary) 'Total Salary' from emp;
```

## Group by and Having clause

**Group by Statement:** The GROUP BY statement is often used with aggregate functions to group the result-set by one or more columns.

```
select dept_no,sum(salary) 'Dept Wise Salary Total' from emp group by  
dept_no;
```

**HAVING Clause:** The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
select dept_no,sum(salary) 'Dept Wise Salary Total' from emp group by dept_no having  
sum(salary)>20000;
```

## MySQL Join

**Simple Join:** A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

```
select e_name, salary, d_name, d_loc from emp, dept where emp.dept_no = dept.dept_no;
```

### Inner Join

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

### Left Join

```
select e.e_name,e.salary,d.d_name,d.d_loc from emp e left join dept d on
e.dept_no=d.dept_no;
```

### Right Join

```
select e.e_name,e.salary,d.d_name,d.d_loc from emp e right join dept d on
e.dept_no=d.dept_no;
```

## Database Programming using PHP

Create a database 'abc' and table 'dept' in MySql

### Function Used:

- **mysql\_connect()** : mysql\_connect() function is used to connect with MySQL database.
- **mysql\_select\_db()**: mysql\_select\_db() is used to create database
- **mysql\_query()** : mysql\_query() is used to execute MySQL query.
- **mysql\_num\_rows()**:It returns the number of rows in a result set.
- **mysql\_fetch\_array()**: It returns row as an associative array. Each key of the array represents the column name of the table or index order.
- **mysql\_close()** : It is used to disconnect with MySQL database.

### Example of Database Programming using PHP

#### Connection Page:

**connection.php:**

```
<?php
```

```

$con = mysql_connect('localhost', 'root', '');
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("abc", $con);
?>

```

Here root -MySQL username  
Password blank

**Insert:** Insert a new record into Department Tables.

**insert.html:**

```

<body>

<form id="form1" name="form1" method="post" action="add.php">
<table width="200" border="0">
<tr><td>Deptno</td><td><input type="text" name="t1" id="t1" /></td></tr>
<tr><td>Dept Name</td><td><input type="text" name="t2" id="t2" /></td></tr>
<tr><td>Location</td><td><input type="text" name="t3" id="t3" /></td></tr>
<tr><td>&nbsp;</td><td><input type="submit" name="b1" id="b1" value="Save"
/></td></tr>
</table>
</form>

```

**Add.php:**

```

<?php
require("connection.php") ;

$dno=$_POST["t1"];
$dname=$_POST["t2"];
$loc=$_POST["t3"];

$q="insert into dept values('.$dno.','.$dname.','.$loc.')";
$r = mysql_query($q);

if(!$r)
{
    echo "Record insertion Failed";
    die(mysql_error());
}
else
    echo "Record inserted successfully !";
mysql_close($con);
?>

```

**OUTPUT:**

|           |                                      |
|-----------|--------------------------------------|
| Deptno    | <input type="text" value="50"/>      |
| Dept Name | <input type="text" value="IT"/>      |
| Location  | <input type="text" value="Kolkata"/> |
|           | <input type="button" value="Save"/>  |

Record inserted successfully !

**Display all records:** Display all department details.

**display.php:**

```
<?php
require("connection.php");
$q="select * from dept";
$result = mysql_query($q);
$count=mysql_num_rows($result);
if($count!=0)
{
?>
<h1>Detail Report</h1>
<table border='1'>
<tr>
<th>Deptno</th>
<th>Dept name</th>
<th>Location</th>
</tr>
<?php
    while($row = mysql_fetch_array($result))
    {
        $dno=$row[0];
        $dname=$row[1];
        $loc=$row[2];
?>
        <tr>
        <td> <?php echo $dno ?></td>
        <td> <?php echo $dname ?></td>
```

```

        <td> <?php echo $loc ?></td>
    </tr>
    <?php
    }
}
else
{
    echo "Empty table";
}
?>
</table>
<?php
mysql_close($con);
?>

```

**OUTPUT:****Detail Report**

Deptno	Dept name	Location
50	IT	Kolkata

**Update Record:** Update Department record.

**update.php:**

```

<?php
require("connection.php") ;
$sql="select * from dept ";

$result=mysql_query($sql);
?>
<form name='f1' method='post' action='edit.php'>
Select Department:
<select name='s1'>
<option>----Select----</option>
<?php
while($row = mysql_fetch_array($result))
{
    $dno=$row[0];
    $dnm=$row[1];
    ?>
    <option value='<?php echo $dno ?>'> <?php echo $dnm ?></option>

```

```
<?php
}
?>
</select>
<input type='submit' name='b1' value='Edit' />
</form>
<?php
mysql_close($con);
?>
```

**edit.php:**

```
<?php
require("connection.php") ;
$dno=$_POST["s1"];
$sql="select * from dept where deptno=".$dno;
$result = mysql_query($sql);
?>

<form name='f1' method='post' action='save.php'>
<table border=0>
<?php
while($row = mysql_fetch_array($result))
{
    $dno=$row[0];
    $dnm=$row[1];
    $loc=$row[2];
    ?>
    <tr>
    <td>Department No</td>
    <td><input type=text name='t1' value='<?php echo $dno ?>'></td>
    </tr>
    <tr><td>Dept Name </td>
    <td><input type=text name='t2' value='<?php echo $dnm ?>'></td>
    </tr>
    <tr><td>Loaction </td>
    <td><input type=text name='t3' value='<?php echo $loc ?>'></td>
    </tr>
    <?php
}
?>
```

**save.php:**

```
<?php
require("connection.php") ;
```

```
$dno=$_POST["t1"];
$dnm=$_POST["t2"];
$loc=$_POST["t3"];

$sql="update dept set dname='".$dnm."', loc='".$loc.'" where deptno=".$dno;
$r = mysql_query($sql);
if(!$r)
{
    echo "Record updation Failed<br>";
    die(mysql_error());
}
else
{
    echo "Record updated successfully !";
}
mysql_close($con);
?>
```

**OUTPUT:**

Select Department:

Department No

Dept Name

Loaction

Record updated successfully !

**Delete Record:** Delete a record from “Dept” Table.

**del.php:**

```
<html>
<body>
<form name="f1" method="post" action="delete.php"><pre>
Department No:<input type="text" name="t1">
<input type="submit" value="Delete">
</form>
</body>
</html>
```

**delete.php:**

```
<?php
require("connection.php") ;
$dno=$_POST["t1"];

$sql="delete from dept where deptno=".$dno;
$r = mysql_query($sql);
if(!$r)
{
echo "Record deletion Failed";
die(mysql_error());
}
else
echo "Record deleted successfully !";
mysql_close($con);
?>
```

**OUTPUT:**

Department No:

Record deleted successfully !



## AJAX Concepts

**AJAX-(Asynchronous JavaScript and XML)** is a platform independent technique for creating fast and dynamic web pages. This is a combination of JavaScript and XML. The data is transferred in the form of XML. Including both client-side and server-side components, AJAX allow the developer to create web applications which can update data (asynchronously) on the web page without a complete reload of the page. This means that it is possible to update parts of a web page, without reloading the whole page.

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:

**Type first character of the name:**

Enter Name:

Suggestions: Amit, Arohi, Arun

### Example of AJAX

#### Sugession.php:

```
<?php
// Array with names
$a[] = "Amit";
$a[] = "Bikash";
$a[] = "Chameli";
$a[] = "Divya";
$a[] = "Ena";
$a[] = "Farhan";
$a[] = "Goutam";
$a[] = "Hena";
$a[] = "Indra";
$a[] = "Joyita";
$a[] = "Koushani";
$a[] = "Lili";
$a[] = "Nabin";
$a[] = "Oindrila";
$a[] = "Piyali";
$a[] = "Arohi";
$a[] = "Riya";
$a[] = "Biplab";
$a[] = "Dhiman";
```

```

$a[] = "Emon";
$a[] = "Arun";
$a[] = "Sunayna";
$a[] = "Titli";
$a[] = "Madhumita";
$a[] = "Dipta";
$a[] = "Liza";
$a[] = "Soumi";
$a[] = "Soumen";
$a[] = "Rohan";
$a[] = "Vicky";
// get the q parameter from URL
$q = $_REQUEST["q"];
$hint = "";
// lookup all hints from array if $q is different from ""
if ($q != "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint == "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}
// Output "no suggestion" if no hint was found or output correct values
echo $hint == "" ? "No suggestion available" : $hint;
?>
Ajax_sugession.php:
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtHint").innerHTML
this.responseText;
            }
        };
    }
};

```

```

        xmlhttp.open("GET", "sugession.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>

<body>
<p><b>Type first character of the name:</b></p>
<form>
Enter Name: <input type="text" name="p" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>

```

**N.B:**

**XMLHttpRequest:** The XMLHttpRequest object can be used to request data from a web server. Update a web page without reloading the page.

**It is used to:**

- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

**XMLHttpRequest Properties:**

- **onreadystatechange:** The onreadystatechange property specifies a function to be executed every time the status of the XMLHttpRequest object changes.

It executes only when we get the response from the server. If we get response from the server means our request is finished which is indicated by 4. And 200 tell us its status that is correct.

- **readyState:** The readyState property defines the current state of the XMLHttpRequest object.
- **readyState-value and description:**
  - 0-The request is not initialized.
  - 1-The request has been set up.
  - 2-The request has been sent.
  - 3-The request is in process.
  - 4-The request is completed.
- **responseText:** The responseText property returns the server response as a text string that can be used to update a web page:

## PHP Filters

PHP filters are used to validate and sanitize data coming from insecure source like user input. It ensures that the application received the correct input. External submitted data (Cookies, Server variables, Database Query Result, user input from a form etc.) can lead to security problems of the web page so you should use filter to validate external data.

### Why use Filter?

Almost all web applications depend on external input. Usually this comes from a user or another application (like a web service). By using filters you can be sure your application gets the correct input type. You should always filter all external data! Input filtering is one of the most important application security issues.

There are two kinds of filters:

**Validating filters:** Validating filters are used to validate the user input. It strict format rules like URL or Email Validating. It returns the expected type on success or false on failure.

**Sanitizing filters:** Sanitizing filters are used to erase any illegal/invalid character from the data.

To filter a variable, use one of the following filter functions:

- **filter\_var()** - Filters a single variable with a specified filter
- **filter\_var\_array()**- Filter several variables with the same or different filters
- **filter\_input()**- Get one input variable and filter it
- **filter\_input\_array()** - Get several input variables and filter them with the same or different filters.

**Sanitize a String:** filter\_var() function to remove all HTML tags from a string.

### Example of Filter

#### Example of Sanitizing

```
<?php
$a = "<font color='purple'>Phoenix Enterprise</font>";
$b = filter_var($a, FILTER_SANITIZE_STRING);
echo $b;
?>
```

#### Example of Validating

##### Validate an Integer:

```
<?php
$val = 0; // $val = 100;
if (filter_var($val, FILTER_VALIDATE_INT) === 0 || !filter_var($val,
FILTER_VALIDATE_INT) === false){
    echo("Valid Integer")."<br>";
}
```

```

else
{
    echo("Not a Valid Integer")."<br>";
}
?>

```

### Sanitize and Validate an Email Address:

```

<?php
$em = "arup.imp@gmail.com";
$em = filter_var($em, FILTER_SANITIZE_EMAIL);
if (!filter_var($em, FILTER_VALIDATE_EMAIL) === false)
{
    echo "Valid email address"."<br>";
}
else
{
    echo "Not a valid email address"."<br>";
}
?>

```

### OUTPUT:

Phoenix Enterprise  
 Valid Integer  
 Valid email address

## Error Handling

**Using die() function:** die( ) function is used to display a message and terminate the execution of the script. die() function prints only string message. Instead of displaying program error it will display customized error message.

**N.B:** The PHP exit( ) function is alias of PHP die( ) function. It is also used to print message.

### Example of die()

```

<?php
$x=10;
$y=10.0;

if($x==$y)
{
    die('both are equal');
    //exit('both are equal');
}
else{
    die('both are not equal');
    //exit('both are not equal') ;
}
echo "<br>End";
?>

```

**OUTPUT:**

**both are equal**

Creating a Custom Error Handler: We can create a custom error handler function that can be called when an error occurs in PHP.

In our example an error occurs if the "age" variable is less than "18":

trigger\_error() : When an application received the incorrect input, it is required to trigger an error. trigger\_error() function is used in that case.

**E\_USER\_WARNING** - This is a Non-fatal user-generated warning using the PHP function trigger\_error()

**Set Error Handler:**

```
set_error_handler("Error",E_USER_WARNING);
```

In this example we are going to use custom error Handler function Error() to handle all errors. set\_error\_handler() function is used to call error handler function. Here First parameter represent custom error handler function name and second parameter could be added to specify an error level.

**Example of Custom Error Handler**

```
<?php
//error handler function
function Error($eno, $str)
{
    echo "<b>Error:</b> [$eno] $str<br>";
    echo "Custome Error Testing";
}

//set error handler
set_error_handler("Error",E_USER_WARNING);

//trigger error
$age=15;
if ($age<=18) {
    trigger_error("Not Eligible for vote",E_USER_WARNING);
}
?>
```

**OUTPUT:**

Error: [512] Not Eligible for vote  
Custome Error Testing

## OOPS Concepts

**Introduction:** In Object Oriented Programming, we design our processes using objects. Therefore, an object-oriented application consists of related objects that group with each other to solve the problem.

**Object Oriented Programming** concept is use to make powerful, **robust** and **secure** programming instructions.

The basic building blocks of object-oriented programming are classes and objects.

### Advantages of OOPs:

- Reusable of code.
- Object-oriented systems can be easily upgraded
- Real world programming.
- Ability to simulate real world event efficiently.
- Information hiding.
- Software complexity can be easily managed.
- It is quite easy to partition the work in a project based on object

## Class and Object Concepts

**Objects:** Objects are the basic run time entities in an object-oriented system. The fundamentals idea behind an object-oriented language is to combine both data and the functions that operate on the data into a single unit known as **object**. In short an entity that has state and behavior is known as an **object** e.g. chair, bike, marker, pen, table, car etc.

In other words Object is the instance of the class. Object can be created by using “**new**” keyword. Class members (**methods and properties**) can directly be accessed through the object using Connector(->) symbol.

### How to create object:

```
$obj= new A();
```

### How to accessed class members using object:

```
$obj->m1();
```

**Class:** The entire set of data and code of an object can be made a user-defined data type with the help of a class. A class is a collection of objects of similar types that means we can create any number of objects belonging to that class.

**Class** is the collection of related data members (properties/variable) and methods (function).

**Define a class:** Class always start with “**class**” keyword and then write class name.

**Syntax:**

```
class Class_Name
{
    //body part
}
```

**Example of Class and Object**

```
<?php
class Demo
{
    var $msg="Hello";
    function add()
    {
        $x=800;
        $y=200;
        $sum=$x+$y;
        echo "sum of given no=".$sum."<br>";
    }

    function sub()
    {
        $x=1000;
        $y=500;
        $sub=$x-$y;
        echo "Sub of given no=".$sub."<br>";
    }

    function multi($a,$b)
    {
        $m=$a*$b;
        echo "Multiplication: ".$m."<br>";
    }
}
$obj= new Demo();
//Connector symbol used to connect objects to their properties or methods.
$obj->add();
$obj->sub();
$obj->multi(5,7);
echo $obj->msg;
?>
```



**OUTPUT:**

```
sum of given no=1000
Sub of given no=500
Multiplication:35
Hello
```

**\$this Keyword**

The \$this is used to indicate the class's own methods and properties, and allows us to have access to them within the class's scope.

**Syntax:**

```
$this -> propertyName;
$this -> methodName();
```

**Example of \$this Keyword**

```
<?php
class A
{
    var $f=1000;
    var $s=500;
    function add( )
    {
        $add=$this->f + $this->s;
        //$add=$first+$second;
        echo "addition=".$add."<br/>";
    }
    function sub( )
    {
        $sub=$this->f - $this->s;
        echo "subtraction=".$sub;
    }
}
$obj= new A();
$obj->add();
$obj->sub();
?>
```

**OUTPUT:**

```
Addition=1500
Subtraction=500
```

## Constructor and destructor in PHP

**Constructor:** Constructor is a special type of method that is used to initialize the object. Constructor is invoked at the time of object creation. It constructs the values i.e. it provides data for the object that is why it is known as **Constructor**. Constructor must have no explicit return type.

In PHP there are two types of Constructor:

**User Defined Constructor:** If a class name and function name will be similar in that case function is considered as **User Defined Constructor**.

### Example of User Defined Constructor

```
<?php
class A
{
    function A()
    {
        echo "User defined constructor"."<br>";
    }

    function m1()
    {
        echo "This is test method of class A"."<br>";
    }
}
$obj= new A();
$obj->m1();
?>
```

### Predefine Constructor:

PHP introduce **\_\_construct()** to define a constructor is known as predefined constructor.

If we change class name then user defined constructor treated as normal method so it is better to use predefined constructor.

### Example of Predefine Constructor

```
<?php
class A
{
    function __construct()
    {
        echo "This is predefined constructor"."<br>";
    }
}
```

```
function m1()
{
    echo "This is test method of class A."<br>";
}
$obj= new A();
$obj->m1();
?>
```

**Note:** if predefined constructor and user defined constructor, both define in the same class, then predefined constructor behave like a Constructor while user defined constructor treated as normal method.

### Example of Constructor

```
<?php
class A
{
    function A()
    {
        echo "User defined constructor."<br>";
    }

    function __construct()
    {
        echo "This is predefined constructor."<br>";
    }

    function m1()
    {
        echo "This is test method of class A."<br>";
    }
}
$obj= new A();
$obj->m1();
$obj->A();
?>
```

**Parameterized Constructor:** A constructor that has parameters is known as parameterized constructor.

### Example Parameterized Constructor

```
<?php
class Employee
{
    public $name;
```

```
public $profile;
function __construct($n,$p)
{
    $this->name=$n;
    $this->profile=$p;
    echo "Welcome ";
}
function show()
{
    echo $this->name."... ";
    echo "Your profile is ".$this->profile."<br/>";
}
}
$obj= new Employee("Amit","Web Designer");
$obj->show();
$obj1= new Employee("Sumit","Developer");
$obj1->show();
?>
```

**OUTPUT:**

Welcome Amit... Your profile is Web Designer

Welcome Sumit... Your profile is Developer

**Destructor:** PHP destructor allows you to clean up resources before PHP releases the object from the memory. Destructors automatically call at last.

To add a destructor to a class, `__destruct()` is used.

**Syntax:**

```
public function __destruct(){
    // clean up resources here
}
```

**Example of Destructor**

```
<?php
class A
{
    function __construct()
    {
        echo "Object is initialized."<br/>;
    }
    function m1()
    {
```

```

        echo "Working " . "<br/>";
    }
    function __destruct()
    {
        echo "Object destroyed automatically after completion of the
work" . "<br>";
    }
}
$obj = new A();
$obj->m1();
echo "Destroyed: " . is_object($obj) . "<br>"; //to check object is destroyed or
not
?>

```

### OUTPUT:

```

Object is initialized
Working
Destroyed: 1
Object destroyed automatically after completion of the work

```

## Access Modifiers

PHP access modifiers are used to set access rights with classes and their members(variables, methods) that are defined within the class scope.

There are three types of Access modifiers:

- public
- protected
- private

**public:** Those class properties and class methods which are set to be public are accessed from inside as well as outside the class definition.

**private:** Class members which are set to be private can only be accessible within the class that defines it.

**protected:** protected is only accessible within the class in which it is defined and its parent or inherited classes.

### Example of Access Modifiers

```

<?php
class parents
{
    public $name="Amit";
    protected $profile="Developer";
    private $salary=80000;
}

```

```

    public function m1()
    {
        echo "Welcome : ".$this->name."<br/>";
        echo "Profile : ".$this->profile."<br/>";
        echo "Salary : ".$this->salary."<br/>";
    }
}
class childs extends parents
{
    public function m2()
    {
        echo "Welcome : ".$this->name."<br/>";
        echo "Profile : ".$this->profile."<br/>";
        echo "Salary : ".$this->salary."<br/>";
    }
}
$obj= new childs;
$obj->m1();
$obj->m2();
$x= new childs;
$x->name="Sumit";
//$x->profile="Web Designer";
$x->salary=30000;
?>

```

**OUTPUT:**

```

Welcome : Amit
Profile : Developer
Salary : 80000
Welcome : Amit
Profile : Developer

```

(!) Notice: Undefined property: childs::\$salary in C:\wamp\www\OOPS\access_sp.php on line 28			
Call Stack			
#	Time	Memory	Function
1	0.0013	373728	{main}()
2	0.0014	374088	childs->m2()

Salary :

## Inheritance

Inheritance provided mechanism that allowed a class to inherit property of another class. It allows you to create a new class that reuses the properties and methods from an existing class.

You can set the common properties and methods in the parent class and specific properties and methods in the child classes. This helps you avoid duplicate code that implements the same properties and method in multiple classes.

A class that inherits from another class is called child class (or sub class or derived class). The class from which the subclass inherits is known as parent class (or superclass or a based class).

We declare a new class with additional keyword extends.

There are three types of inheritance like single, multiple and multi-level inheritance. PHP supports single and multi-level inheritance. It will not support multiple inheritances.

### Example of Inheritance

#### Single Inheritance

```
<?php
class A
{
    function m1($x,$y)
    {
        $r=$y*$y;
        echo "Multiplication = ".$r."<br>";
    }
}

class B extends A
{
    function m2($x,$y)
    {
        $r=$x+$y;
        echo "Sum = ".$r."<br>";
    }
}

$obj= new B();
$obj->m1(1000,500);
$obj->m2(1000,500);
?>
```

**OUTPUT:****Multiplication = 250000****Sum = 1500****Multilevel Inheritance**

```
<?php
class BaseClass
{
    function add()
    {
        $x=1000;
        $y=500;
        $add=$x+$y;
        echo "Addition=".$add."<br/>";
    }
}
class child extends BaseClass
{
    function sub()
    {
        $x=1000;
        $y=500;
        $sub=$x-$y;
        echo "subtraction=".$sub."<br/>";
    }
}
class Nestedchild extends child
{
    function mult()
    {
        $x=1000;
        $y=500;
        $mult=$x*$y;
        echo "multiplication=".$mult."<br/>";
    }
}

class show extends Nestedchild
{
    function __construct()
    {
        parent::add();
        parent::sub();
        parent::mult();
    }
}
```



```
    }  
}  
$obj= new show();  
?>
```

**OUTPUT:**

**Addition=1500  
subtraction=500  
multiplication=500000**

## Method Overriding

When a method in a sub class has same name and type signature as a method in its super class, then the method is known as overridden method.

**Example of Method Overriding**

```
<?php  
class A  
{  
    function m1($x,$y)  
    {  
        $r=$y*$y;  
        echo "Multiplication = ".$r;  
    }  
}  
class B extends A  
{  
    function m1($x,$y)  
    {  
        $r=$x+$y;  
        echo "Sum  = ".$r;  
    }  
}  
$obj= new B();  
$obj->m1(1000,500);  
?>
```

**OUTPUT:**

**Sum = 1500**

## Encapsulation and Abstraction

Encapsulation is a concept of wrapping up related data members (properties and methods) in a single unit known as encapsulation and hiding the essential internal property of that unit known as Data abstraction.

### Example of Encapsulation

```
<?php
class person
{
    public $name;
    public $age;
    function __construct($n, $a)
    {
        $this->name=$n;
        $this->age=$a;
    }
    public function setAge($ag)
    {
        $this->ag=$ag;
    }
    public function display()
    {
        echo "welcome ".$this->name."<br/>";
        return $this->age-$this->ag;
    }
}
$person=new person("Amit",30);
$person->setAge(50);
echo "You are ".$person->display()." years old";
?>
```

### OUTPUT:

```
welcome Amit
You are -20 years old
```

**Explanation:** In the above example Two properties \$name and \$age along with the methods **setAge()** and **display()** are defined in a single unit (class person). We create the object of class person and that object called the methods and methods are performed their task with defined properties.

## Final keyword

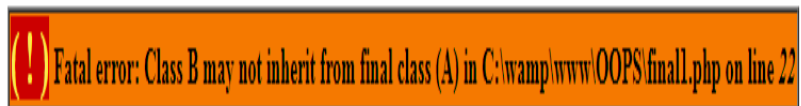
In PHP, final keyword is applicable only for classes and methods.

If you make any class as final then the class cannot be inherited by some other class.

### Example of final class

```
<?php
final class A
{
    function m1()
    {
        echo "Inside m1()";
    }
}
class B extends A
{
    function m2()
    {
        echo "Inside m2()";
    }
}
$obj = new B();
$obj->m1();
?>
```

### OUTPUT:



- If you define a method with final keyword then the method cannot be overridden.

### Example of final method

```
<?php
class A
{
    final function show($x,$y)
    {
        $sum=$x+$y;
        echo "Sum of given no=".$sum;
    }
}
class B extends A
```

```

{
    function show($x,$y)
    {
        $mult=$x*$y;
        echo "Multiplication of given no=".$mult;
    }
}
$obj= new B();
$obj->show(100,100);
?>

```

**OUTPUT:**


Fatal error: Cannot override final method A::show() in C:\wamp\www\OOPS\final.php on line 24

## PHP Abstract class

Abstract classes are classes that contain at least one abstract method which must be declared not defined contains only signature and require subclasses to provide implementations for the abstract methods. Basically it is a technique of hiding information of the class. You cannot create an object or instance of an abstract class.

There must be an **“abstract”** keyword that must be return before this class for it to be abstract class.

### Example of Abstract Class

```

<?php
abstract class A
{
    abstract function m1();
    public function m2()
    {
        echo "Inside m2 function";
    }
}
class B extends A
{
    public function m1()
    {
        echo "Inside m1 function<br/>";
    }
}
$obj = new B();
$obj->m1();
$obj->m2();
?>

```

**OUTPUT:****Inside m1 function****Inside m2 function****Interface**

An interface is a blueprint of a class. It has abstract methods and final fields. All methods of the interface are abstract methods this means that interfaces do not specify any code to implement these methods. It is the responsibility of that class that implements an interface to define the code for implementation of these methods. An interface can have its own constants. It is mainly used to achieve full abstraction. To define an interface, you use the interface keyword.

**Example of Interface**

```
<?php
interface A
{
    function f1();
}
interface B
{
    function f2();
}
class C implements A,B
{
    function f1()
    {
        echo "Phoenix"."<br>";
    }
    function f2()
    {
        echo "TCS iON"."<br>";
    }
}
$obj=new C();
$obj->f1();
$obj->f2();
?>
```

**OUTPUT:****Phoenix****TCS iON**

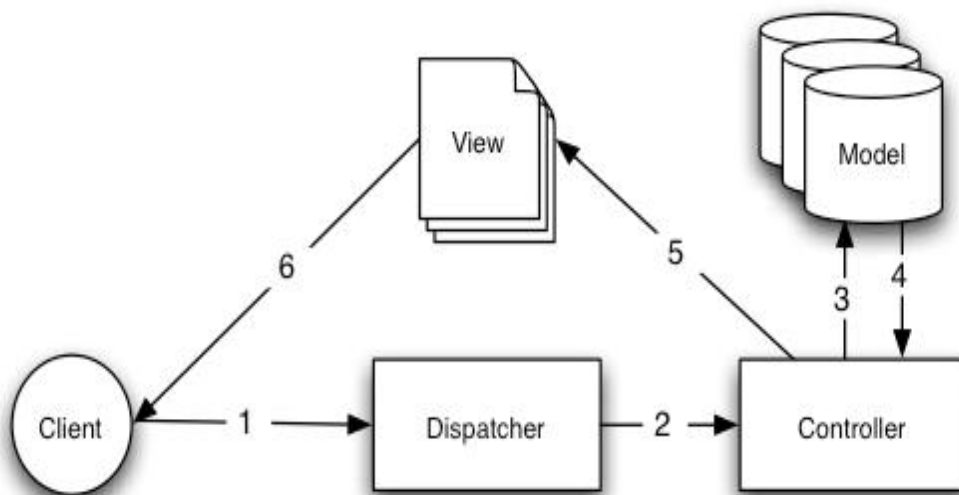
## MVC

### Why use MVC?

Because it is a true software design pattern that separate an application into models, views, and controllers makes your application very light. New features are easily added, and new faces on old features are a snap. The modular and separate design also allows developers and designers to work simultaneously. It also allows developers to make changes in one part of the application without affecting the others.

The typical program flow in MVC is:

- The model, view and controller are initialized.
- The view is displayed the interface to the Client, reading data from the model
- The Client interacts with the view (e.g. presses a button) which calls a specified controller action
- The controller dispatches the user inputs to the model for processing
- The model processes data and through controller dispatches the output to the view
- The view is refreshed (fetching the updated data from the model)



### Example of MVC

```
<?php
class Model
{
    public $data;
    public function __construct()
    {
        $this->data='Hello World... !';
    }
}
class View
{
    private $m;
    public function __construct(Model $mod)
    {
        $this->m=$mod;
    }
    public function display()
    {
        echo '<h1>'.$this->m->data.'</h1>'; }
}
class Controller
{
    private $m;
    public function __construct(Model $model)
    {
        $this->m=$model;
    }
}
$Model = new Model(); //Creating object of model class, The data will be
initialized
$Controller = new Controller($Model); //Calling parameterized constructor of
controller class
$View = new View($Model); //Calling parameterized constructor of view class
echo $View->display(); //Calling display() method of view class
?>
```

First, we are creating an object of **Model** class.

Next, we are creating an object of **View** class View class by calling parameterized constructor of **View** class.

With the help of this object we are calling **display()** which shows the output i.e the actual data of **Model** Class.

Notice how the controller isn't doing anything? This is because there is no user interaction. Hello world is directly display.

Extend the above Hello World example and make it so that when "Hello world" is clicked, the text will be changed to "CMC Academy".

Firstly, add a link to the view. This can be likened to adding a callback to the controller from the view. In GUI applications, the controller would just be another object in memory and the callback would be a simple function call. This is not an option on the web the only way to access the controller is another HTTP request.

```
<?php
class Model
{
    public $data;
    public function __construct()
    {
        $this->data='Hello World ..... !';
    }
}
class View
{
    private $m;
    public function __construct(Model $mod)
    {
        $this->m=$mod;
    }
    public function display()
    { echo '<a href="mvc2.php?action=textclicked">' . $this->m->data . '</a>';
    }
}
class Controller
{
    private $m;
    public function __construct(Model $model)
    {
        $this->m=$model;
    }
    public function textClicked()
    {
        $this->m->data = 'CMC Academy';
    }
}
$Model = new Model(); //Creating object of model class, The data will be
initialized
$Controller = new Controller($Model); //Calling parameterized constructor of
controller class
$View = new View($Model); //Calling parameterized constructor of view class
```



```

if (isset($_GET['action']))
{
$Controller->{$_GET['action']}();
}
echo $View->display(); //Calling display() method of view class
?>

```

**Model:** Model manage the data, it stores and retrieve the data usually from the data base.

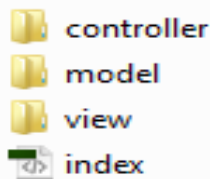
**View:** View represents the data in the required format. **Controller:** it is handle the Model and View layer. Controller take the request from the client send to Model for operation and then send the required output to View.

**Now let start to create the simple login form in php using MVC pattern.**

### Step 1:

Make a new folder named SmartPHP (name can be customised) under www of WampServer. In this folder make 3 folders with the name controller, model, view respectively and also create a PHP page with a name index.php.

Directory Structure of SmartPHP



### Step 2:

In the controller folder create a PHP page with the name Controller.php and write the following

**code:**

```

<?php
include_once("model/Model.php");
class Controller
{
public $m;
public function __construct()
{
$this->m = new Model();
}
public function invoke()
{
$result = $this->m->login(); //calling login() function of model class and
//store the return value in $result
if($result == 'Correct')
include 'view/home.php';
else
include 'view/login.php';
}
}

```

```
}
?>
```

**Step 3:**

In the model folder create a PHP page with the name Model.php and write the following code:

```
<?php
class Model
{
public function login()
{
$r='';
if(isset($_REQUEST['uid']) && isset($_REQUEST['pas']))
{
if($_REQUEST['uid']=='hello' && $_REQUEST['pas']=='world') $r= 'Correct';
else
$r= 'Incorrect Userid or Password';
}
return $r;
}
}
?>
```

**Step 4:**

In the view folder create two(2) PHP page with the name login.php, home.php respectively. and write the following code:

**login.php**

```
<?php
echo $result;
?>
<form action="" method="POST">
<table width="283" border="0">
<tr>
<td>Username</td>
<td><input id="username" value="" name="uid" type="text" required="required"
/></td>
</tr>
<tr>
<td>Password</td>
<td><input id="password" name="pas" type="password" required="required"
/></td>
</tr>
<tr>
<td>&nbsp;</td>
<td><input type="submit" name="b1" id="b1" value="Login" />
```

```
<input type="reset" name="b2" id="b2" value="Reset" /></td>
</tr>
</table>
</form>
</body>
</html>
```

### home.php

```
<?php
echo "<p>H!";
echo "<p>Thank you, you have logged in Successfully";
?>
```

**Step 5:** Open the **index.php** file and write the following code:

```
<?php
include_once("controller/Controller.php");
$control = new Controller();
$control->invoke();
?>
```

**Explanation of above code structure:** Starting from index.php

```
<?php
include_once("controller/Controller.php");
$control = new Controller();
$control->invoke();
?>
```

- Including Controller.php under controller folder
- Creating an object called control
- Invoking invoke() method of class controller

```
<?php
include_once("model/Model.php");
class Controller
{
    public $m;
    public function __construct()
    {
        $this->m = new Model();
    }
    public function invoke()
    {
        $result = $this->m->login();
        if($result == 'Correct')
            include 'view/home.php';
    }
}
```

```

else
include 'view/login.php';
}
}
?>

```

The constructor of controller class is creating an object of Model Class \$m

- The **invoke()** method invoked by controller(As mention above) in turn calling **login()** of Model class.

```

public function login()
{
$r='';
if(isset($_REQUEST['uid']) && isset($_REQUEST['pas']))
{
if($_REQUEST['uid']=='hello' && $_REQUEST['pas']=='world') $r= 'Correct';
else
$r= 'Incorrect Userid or Password';
}
return $r;
}

```

For the first time of execution as it won't get any user request body if statement will not be executed. So it will return blank result to invoke() method of controller class.

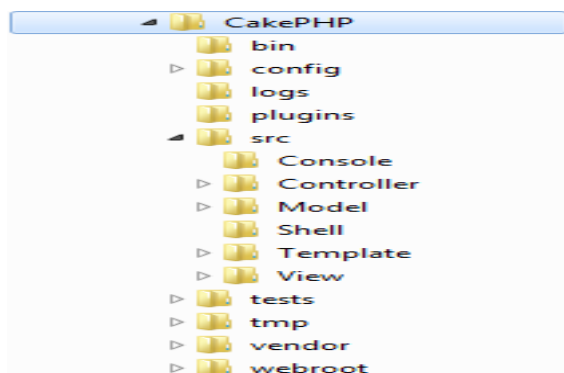
- Therefore else part of invoke() method will include **login.php** of view folder
- After getting user input from login.php the controller will select view.

## Cake PHP

Installing CakePHP Download it from github – <https://github.com/cakephp/cakephp/releases>

Create a folder called CakePHP (You can any give custom name to folder) under www of wamp. After downloading it from github, extract all the files in the folder CakePHP under www of WampServer.

After extracting it, the folder structure will look like this:



Start Wamp Server. Let's check whether it has been installed correctly or not by visiting the following URL in browser – <http://localhost/CakePHP>

### Folder Name & Description

**config** : The config folder holds the (few) configuration files CakePHP uses. Database connection details, bootstrapping, core configuration files and more should be stored here.

**src** : CakePHP's src folder is where you will do most of your application development. Let's look a little closer at the folders inside src.

- **Controller** Contains your application's controllers and their components.
- **Model** Contains your application's tables, entities and behaviors.
- **View** Presentational classes are placed here: cells, helpers, and template files.
- **Template** Presentational files are placed here: elements, error pages, layouts, and view template files.

CakePHP comes with one configuration file **routes.php** under a dedicated folder **config** by default and we can modify it according to our needs.

Following Example shows how to implement a web application using Cake PHP.

### Example of Cake PHP:

#### Step 1:

Make changes in the **config/routes.php** file as shown in the following program.

```
<?php
use Cake\Core\Plugin;
use Cake\Routing\RouteBuilder;
use Cake\Routing\Router;
Router::defaultRouteClass('DashedRoute');
Router::scope('/', function (RouteBuilder $routes) {
    $routes->connect('extend', ['controller'=>'Extends', 'action'=>'index']);
    $routes->fallbacks('DashedRoute');
});
Plugin::routes();
?>
```

#### Step 2:

Create a **ExtendsController.php** file at **src/Controller/ExtendsController.php**. Copy the following code in the controller file.

**src/Controller/ExtendsController.php**

```
<?php
namespace App\Controller;
use App\Controller\AppController;
class ExtendsController extends AppController{
public function index(){
}
}
?>
```

This is an example of extending view.

Finally:

Execute the above example by visiting the following URL.

<http://localhost/CakePHP/extend>

## Learner's Sample Assignment

### Login System:

We create a login form where user can enter their userid and password. When user submit the form these inputs will be verified against the credentials stored in the database, if the userid and password match, the user is authorized and granted access to the site, otherwise the login attempt will be rejected.

**Step 1:** Creating the Database Table-Execute the following SQL query to create the “profile” table inside your MySQL database “online\_exam\_system” .

```
CREATE TABLE users ( user_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, st_name
VARCHAR(50) NOT NULL, password VARCHAR(255) NOT NULL, user_type VARCHAR(255)
NOT NULL );
```

Insert minimum one record for admin user and one record for normal user in your table.

**Step 2:** Connecting MySQL- To connect PHP application with the MySQL database let's create a file named "**connection.php**" and place the following code inside it.

### connection.php: (Connection Page)

```
<?php
$con = mysql_connect('localhost', 'root', '');
if (!$con)
{
die('Could not connect: ' . mysql_error());
}
mysql_select_db("online_exam_system", $con);
```

?>

**Step 3:** Creating the Login Form-Let's create a file named "login.php" and place the following code inside it.

**login.php:**

<?php

```
require("connection.php");
session_start();
if(isset($_POST['b1']))
{
    $uid=$_POST["uname"];
    $pwd=$_POST["pwd"];
    $epwd=md5($pwd); //encrypted format

    $q="select user_id,user_type,st_name from profile,student where
    user_id='".$uid.'" and password='".$epwd.'" and
    student.st_id=profile.user_id";

    $r=mysql_query($q);
    if(!$r)
        die(mysql_error());
    else
    {
        if($row=mysql_fetch_array($r))
        {
            $sname=$row[2];
            $utype=$row[1];
            echo $utype;
            if($utype=="ADMIN")
            {
                $_SESSION["uid"]=$uid;
                $_SESSION["name"]="Administrator";
                $_SESSION["utype"]=$utype;
                header('Location:admin_home.php');
            }
            else
            {
                $_SESSION["name"]=$sname;
                $_SESSION["uid"]=$uid;
                $_SESSION["utype"]=$utype;
                header('Location:student_home.php');
            }
        }
        else
        {
            echo "Invalid User id and password";
        }
    }
}
```





## Sample Mini Project Title

### Online Auction System

Online Auction however is a different business model where the items are sold through price bidding. Usually bidding have start price and ending time. Potential buyers in auction and the winner is the one who bids the item for highest price. We treat the fraud detection with a binary classification. For buying product online user have to provide his personal details like email address, license number, PAN number etc. Only the valid user will have authority to bid. This prevents various frauds according in online shopping.

### Daily Expense Tracker System

Daily Expense Tracker System is designed to keep a track of Income-Expense of a Housewife on a day-to-day basis. This System divides the Income based on daily expenses. If you exceed day's expense, system will cut it from your income and will provide new daily expense allowed amount. If that day's expense is less, system will add it in savings. Daily expense tracking System will generate report at the end of month to show Income-Expense Curve. It will let you add the savings amount, which you had saved for some particular Festivals or days like Birthday or Anniversary.

### College Social Network

Web application intends to provide a well-established web-based Social Network system between a job seeker and a recruiter. This documents a networking system scope, functionalities, requirements and feasibility. This project aims to develop a website which provides a Communication among peoples on network, which works quite similar to Social Media Site. This website also provides the features of writing and posting a post or any event all at one place. The main idea behind it is to share the job related details posted by placement officer via adding a post which can be read by all the student as well as faculty using the website. This web application can be handled by the admin and manage student as well as faculty.