Chapter 1: Introduction to PHP

What is PHP?

PHP is a general-purpose scripting language especially suited to web development.

- PHP stands for PHP hypertext preprocessor
- PHP is a server side programming language that is embedded in HTML.
- PHP allows web developers to create dynamic contents that interact with database, session management etc.
- It is integrated with popular databases like MySQL, postgreSQL, Oracle etc.
- PHP is basically used for developing web apps
- PHP is open source

Why PHP?

- PHP runs on different OS platforms like
  - Windows (WAMP)
  - Linux (LAMP)
  - MAC OS (MAMP)
- PHP is an interpreted language.
- PHP is compatible with almost all servers but the most used server is apache.
- PHP is open source so it is free to download and use
- PHP is easy to use and widely used.

Characteristics of PHP

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Who are using PHP?

Many reputed domains are using PHP

- Facebook
- Yahoo
- Wikipedia
- WordPress

20+ million other domains are using PHP

History of PHP

Versions:

- PHP /FI 1.0 released in 1995
- PHP /FI 2.0 released in 1997
- PHP 3.0 released in 1998

- PHP 4.0 released in 2000
- PHP 5.0 released in 2004
- PHP 5.3 released in 2009
- PHP 7.0 released in 2014

Advantages of PHP:

- Most important advantage of PHP is that it’s open source and free from cost. It can be downloaded at anywhere and readily available to use for event of web applications.
- It is platform independent. PHP based applications can run on any OS like UNIX, Linux and Windows, etc.
- Application can easily be loaded which are based on PHP and connected to database. It’s mainly used due to its faster rate of loading over slow internet speed than another programming language.
- It has less learning curve, because it is simple and straightforward to use. Someone familiar with C programming can easily work on PHP.
- It is more stable from a few years with assistance of providing continuous support to various versions.
- It helps in reusing an equivalent code and no got to write lengthy code and sophisticated structure for event of web applications.
- It helps in managing code easily.
- It has powerful library support to use various function modules for data representation.
- PHP’s built-in database connection modules help in connecting database easily reduce trouble and time for development of web applications and content based sites.
- Popularity of PHP gave rise to various community of developers, a fraction of which may be potential candidates for hire.
- Flexibility makes PHP ready to effectively combine with many other programming languages in order that the software package could use foremost effective technology for every particular feature.

Syntax of PHP:

A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>

```
<?php
// PHP code goes here
?>
<?php
echo "Hello, World!";
?>
```

echo is a command used in PHP to display anything on screen.

Comments:

- // this is single line comment
- # this is also single line comment
- /* this is multi line comment,
- can spread over thousands of lines */

Variables

Variable is name given to memory location whose value may vary during the execution of program

Variables in a program are used to store some values or data that can be used later in a program.

- Any variables declared in PHP must begin with a dollar sign ($), followed by the variable name.
- A variable can have long descriptive names (like $factorial, $even_nos) or short names (like $n or $f or $x)
- A variable name can only contain alphanumeric characters and underscores (i.e., 'a-z', 'A-Z', '0-9 and '_') in their name. Even it cannot start with a number.
- Assignment of variables is done with the assignment operator, "equal to (=)". The variable names are on the left of equal and the expression or values are to the right of the assignment operator '='.
- One must keep in mind that variable names in PHP names must start with a letter or underscore and no numbers.
- PHP is a loosely typed language, and we do not require to declare the data types of variables, rather PHP assumes it automatically by analyzing the values. The same happens while conversion. No variables are declared before they are used. It automatically converts types from one type to another whenever required.
- PHP variables are case-sensitive, i.e. $sum and $SUM are treated differently.

Constants

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Syntax:

```
define(name, value, case-insensitive)
```

example:

```
<!DOCTYPE html>
<html>
<body>
<?php
// case-sensitive constant name
define("GREETING", "Welcome to Softanic!");
echo GREETING;
?>
</body>
</html>
```

Variable Scopes
Scope of a variable is defined as its extent in program within which it can be accessed, i.e. the scope of a variable is the portion of the program within which it is visible or can be accessed.
Depending on the scopes, PHP has three variable scopes:
Local variables: The variables declared within a function are called local variables to that function and has its scope only in that particular function. In simple words, it cannot be accessed outside that function. Any declaration of a variable outside the function with same name as that of the one within the function is a completely different variable. We will learn about functions in detail in later articles. For now, consider a function as a block of statements.
Global variables: The variables declared outside a function are called global variables. These variables can be accessed directly outside a function. To get access within a function we need to use the "global" keyword before the variable to refer to the global variable.
Static variable: It is the characteristic of PHP to delete the variable, ones it completes its execution and the memory is freed. But sometimes we need to store the variables even after the completion of function execution. To do this we use static keyword and the variables are then called as static variables. PHP associates a data type depending on the value for the variable.

```
<?php
function static_var()
{
	// static variable
	static $num = 5;
	$sum = 2;

	$sum++;
	$num++;

	echo $num, "\n";
	echo $sum, "\n";
}
// first function call
static_var();

// second function call
static_var();

?>
Output:

6
3
7
3
```

Datatypes

- Data types specify the size and type of values that can be stored.
- Variable does not need to be declared ITS DATA TYPE adding a value to it.
- PHP is a Loosely Typed Language so here no need to define data type
- To check only data type use gettype( ) function.
- To check value, data type and size use var_dump( ) function.

Various data type:

- Integers
- Doubles
- NULL
- Strings
- Booleans
- Arrays
- Objects
- Resources

Expressions and operators

Operators are symbols that tell the PHP processor to perform certain actions. For example, the addition (+) symbol is an operator that tells PHP to add two variables or values, while the greater-than (>) symbol is an operator that tells PHP to compare two values.

The following lists describe the different operators used in PHP.

PHP Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc.
Here's a complete list of PHP's arithmetic operators:

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Arithmetic Operators</title>
</head>
<body>
<?php
$x = 10;
$y = 4;
echo($x + $y);
echo "<br>";
echo($x - $y);
echo "<br>";
echo($x * $y);
echo "<br>";
echo($x / $y);
echo "<br>";
echo($x % $y);
?>
</body>
</html>
```

Output:

```
14
6
40
2.5
2
```

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Operator	Description	Example	Is The Same As
=	Assign	$x = $y	$x = $y
+=	Add and assign	$x += $y	$x = $x + $y
-=	Subtract and assign	$x -= $y	$x = $x - $y
*=	Multiply and assign	$x *= $y	$x = $x * $y
/=	Divide and assign quotient	$x /= $y	$x = $x / $y
%=	Divide and assign modulus	$x %= $y	$x = $x % $y

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Assignment Operators</title>
</head>
<body>

<?php
$x = 10;
echo $x;
echo "<br>";
$x = 20;
$x += 30;
echo $x;
echo "<br>";
 $x = 50;
```

```
$x -= 20;
echo $x;
echo "<br>";
 $x = 5;
$x *= 25;
echo $x;
echo "<br>";
 $x = 50;
$x /= 10;
echo $x;
echo "<br>";
 $x = 100;
$x %= 15;
echo $x;
?>
</body>
</html>
```

Output:

```
10
50
30
125
5
10
```

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
<	Less than	$x < $y	True if $x is less than $y

>	Greater than	$x > $y	True if $x is greater than $y
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Comparison Operators</title>
</head>
<body>
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z);
echo "<br>";
var_dump($x === $z);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x !== $z);
echo "<br>";
var_dump($x < $y);
echo "<br>";
var_dump($x > $y);
echo "<br>";
var_dump($x <= $y);
echo "<br>";
var_dump($x >= $y);
?>
</body>
</html>
```

Output:

```
bool(true)
bool(false)
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)
bool(false)
```

PHP Incrementing and Decrementing Operators

The increment/decrement operators are used to increment/decrement a variable's value.

Operator	Name	Effect
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Incrementing and Decrementing Operators</title>
</head>
<body>
<?php
$x = 10;
echo ++$x;
echo "<br>";
echo $x;
echo "<hr>";
$x = 10;
echo $x++;
echo "<br>";
echo $x;
echo "<hr>";

$x = 10;
echo --$x;
echo "<br>";
echo $x;
echo "<hr>";

$x = 10;
echo $x--;
echo "<br>";
echo $x;
?>
```

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

Output:

```
11
11
________________________________________
10
11
________________________________________
9
9
________________________________________
10
9
```

PHP Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $$x or $y is true
!	Not	!$x	True if $x is not true

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Logical Operators</title>
</head>
<body>

<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 ==
0))){
    echo "$year is a leap year.";
} else{
    echo "$year is not a leap year.";
}
?>

</body>
</html>
```

Output:

```
2014 is not a leap year.
```

PHP String Operators

There are two operators which are specifically designed for strings.

Operator	Description	Example	Result
.	Concatenation	$str1 . $str2	Concatenation of $str1 and $str2
.=	Concatenation assignment	$str1 .= $str2	Appends the $str2 to the $str1

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP String Operators</title>
</head>
<body>
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!
echo "<br>";

$x .= $y;
echo $x; // Outputs: Hello World!
?>
</body>
</html>
```

Output:

```
Hello World!
Hello World!
```

PHP Array Operators

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	True if $x and $y have the same key/value pairs
===	Identity	$x === $y	True if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	True if $x is not equal to $y
<>	Inequality	$x <> $y	True if $x is not equal to $y
!==	Non-identity	$x !== $y	True if $x is not identical to $y

The array operators are used to compare arrays:

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Array Operators</title>
</head>
<body>
<?php
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
$z = $x + $y; // Union of $x and $y
var_dump($z);
```

```
echo "<hr>";
var_dump($x == $y);
echo "<br>";
var_dump($x === $y);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x <> $y);
echo "<br>";
var_dump($x !== $y);
?>
</body>
</html>
```

Output:

```
array(6) { ["a"]=> string(3) "Red" ["b"]=> string(5) "Green"
["c"]=> string(4) "Blue" ["u"]=> string(6) "Yellow" ["v"]=>
string(6) "Orange" ["w"]=> string(4) "Pink" }

bool(false)
bool(false)
bool(true)
bool(true)
bool(true)
```

Decision making control statements

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

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

if Statement

The `if` statement executes some code if one condition is true.

Syntax:

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

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
}
?>
</body>
</html>
```

Output:

```
Have a good day!
```

if...else Statement

The `if...else` statement executes some code if a condition is true and another code if that condition is false.

Syntax:

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

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
</body>
</html>
```

Output:

```
Have a good day!
```

if...elseif...else Statement

The `if...elseif...else` statement executes different codes for more than two conditions.

Syntax:

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if first condition is false and this
condition is true;
} else {
    code to be executed if all conditions are false;
}
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
```

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

Output:

```
The hour (of the server) is 10, and will give the following
message:
Have a good day!
```

switch Statement

The `switch` statement is used to perform different actions based on different conditions.

Syntax:

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

Example:

```
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor
green!";
}
?>

</body>
</html>
```

Output:

```
Your favorite color is red!
```

break Statement

Sometimes a situation arises where we want to exit from a loop immediately without waiting to get back to the conditional statement.
The keyword break ends execution of the current for, foreach, while, do while or switch structure. When the keyword break executed inside a loop the control automatically passes to the first statement outside the loop. A break is usually associated with the if.

Example:

```
<?php
$array1=array(100, 1100, 200, 400, 900);
$x1=0;
$sum=0
while ($x1<=4)
{
if ($sum>1500)
{
break;
}
$sum = $sum+$array1[$x1];
$x1=$x1+1;
}
echo $sum;
?>
```

Output:

```
1800
```

continue Statement

Sometimes a situation arises where we want to take the control to the beginning of the loop (for example for, while, do while etc.) skipping the rest statements inside the loop which have not yet been executed.

The keyword continue allow us to do this. When the keyword continue executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if.
Example:

```
<?php
$x=1;
echo 'List of odd numbers between 1 to 10 <br />';
while ($x<=10)
{
if (($x % 2)==0)
{
$x++;
continue;
}
else
{
echo $x.'<br />';
$x++;
}
}
?>
```

Output:

```
List of odd numbers between 1 to 10
1
3
5
7
9
```

Loop control structure

Loops are used to execute the same block of code again and again, as long as a certain condition is true.
In PHP, we have the following loop types:

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

while Loop

The `while` loop executes a block of code as long as the specified condition is true.
Syntax:

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

Examples:

```
<?php
$x = 1;
while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

Output:

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

Do while Loop

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

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

Examples:

```
<?php
$x = 1;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

Output:

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

for Loop

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

```
for (init counter; test counter; increment counter) {
    code to be executed for each iteration;
}
```

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

Example:

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

Output:

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

Foreach Loop

The `foreach` loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax:

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

Example:

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

Output:

```
red
green
blue
yellow
```

Unit II: Arrays, Functions & Graphics

Creating and manipulating arrays

Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key.

Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to a hundred, then it would be really difficult for the user or developer to create so many different variables. Here array comes into play and helps us to store every element within a single variable and also allows easy access using an index or a key. An array is created using an array() function in PHP.

There are basically three types of arrays in PHP:

Indexed or Numeric Arrays: An array with a numeric index where values are stored linearly.

Associative Arrays: An array with a string index where instead of linear storage, each value can be assigned a specific key.

Multidimensional Arrays: An array which contains single or multiple array within it and can be accessed via multiple indices

Indexed or Numeric Arrays

These type of arrays can be used to store any type of elements, but an index is always a number. By default, the index starts at zero. These arrays can be created in two different ways as shown in the following example:

```
<?php

// One way to create an indexed array
$name_one = array("Zack", "Anthony", "Ram", "Salim", "Raghav");

// Accessing the elements directly
echo "Accessing the 1st array elements directly:\n";
echo $name_one[2], "\n";
echo $name_one[0], "\n";
echo $name_one[4], "\n";

// Second way to create an indexed array
$name_two[0] = "ZACK";
$name_two[1] = "ANTHONY";
$name_two[2] = "RAM";
$name_two[3] = "SALIM";
$name_two[4] = "RAGHAV";

// Accessing the elements directly
```

```
echo "Accessing the 2nd array elements directly:\n";
echo $name_two[2], "\n";
echo $name_two[0], "\n";
echo $name_two[4], "\n";

?>
```

Output:

```
Accessing the 1st array elements directly:
Ram
Zack
Raghav
Accessing the 2nd array elements directly:
RAM
ZACK
RAGHAV
```

Traversing: We can traverse an indexed array using loops in PHP. We can loop through the indexed array in two ways. First by using for loop and secondly by using foreach. You can refer to PHP | Loops for the syntax and basic use.

```
<?php

// Creating an indexed array
$name_one = array("Zack", "Anthony", "Ram", "Salim",
"Raghav");

// One way of Looping through an array using foreach
echo "Looping using foreach: \n";
foreach ($name_one as $val){
      echo $val. "\n";
}

// count() function is used to count
// the number of elements in an array
$round = count($name_one);
echo "\nThe number of elements are $round \n";

// Another way to loop through the array using for
echo "Looping using for: \n";
for($n = 0; $n < $round; $n++){
      echo $name_one[$n], "\n";
}

?>
```

Associative Arrays

These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

```
<?php

// One way to create an associative array
$name_one = array("Zack"=>"Zara", "Anthony"=>"Any",
                      "Ram"=>"Rani", "Salim"=>"Sara",
                      "Raghav"=>"Ravina");

// Second way to create an associative array
$name_two["zack"] = "zara";
$name_two["anthony"] = "any";
$name_two["ram"] = "rani";
$name_two["salim"] = "sara";
$name_two["raghav"] = "ravina";

// Accessing the elements directly
echo "Accessing the elements directly:\n";
echo $name_two["zack"], "\n";
echo $name_two["salim"], "\n";
echo $name_two["anthony"], "\n";
echo $name_one["Ram"], "\n";
echo $name_one["Raghav"], "\n";

?>
```

Output:

```
Accessing the elements directly:
zara
sara
any
Rani
Ravina
```

Traversing Associative Arrays: We can traverse associative arrays in a similar way did in numeric arrays using loops. We can loop through the associative array in two ways. First by using for loop and secondly by using foreach. You can refer to PHP | Loops for the syntax and basic use.

Example:

```
<?php

// Creating an associative array
$name_one = array("Zack"=>"Zara", "Anthony"=>"Any",
                           "Ram"=>"Rani", "Salim"=>"Sara",
                           "Raghav"=>"Ravina");

// Looping through an array using foreach
echo "Looping using foreach: \n";
foreach ($name_one as $val => $val_value){
     echo "Husband is ".$val." and Wife is ".$val_value."\n";
}

// Looping through an array using for
echo "\nLooping using for: \n";
$keys = array_keys($name_two);
$round = count($name_two);

for($i=0; $i < $round; ++$i) {
     echo $keys[$i] . ' ' . $name_two[$keys[$i]] . "\n";
}

?>
```

Output:

```
Looping using foreach:
Husband is Zack and Wife is Zara
Husband is Anthony and Wife is Any
Husband is Ram and Wife is Rani
Husband is Salim and Wife is Sara
Husband is Raghav and Wife is Ravina

Looping using for:
zack zara
anthony any
ram rani
salim sara
raghav ravina
```

Multidimensional Arrays

Multi-dimensional arrays are such arrays that store another array at each index instead of a single element. In other words, we can define multi-dimensional arrays as an array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within. Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

Example:

```
<?php

// Defining a multidimensional array
$favorites = array(
   array(
           "name" => "Dave Punk",
           "mob" => "5689741523",
           "email" => "davepunk@gmail.com",
   ),
   array(
           "name" => "Monty Smith",
           "mob" => "2584369721",
           "email" => "montysmith@gmail.com",
   ),
   array(
           "name" => "John Flinch",
           "mob" => "9875147536",
           "email" => "johnflinch@gmail.com",
   )
);

// Accessing elements
echo "Dave Punk email-id is: " . $favorites[0]["email"], "\n";
echo "John Flinch mobile number is: " . $favorites[2]["mob"];

?>
```

Output:

```

Dave Punk email-id is: davepunk@gmail.com
John Flinch mobile number is: 9875147536
```

Array functions

1. **Implode()**

PHP implode Function joins elements of an array into a single string. It returns a string containing elements of the array.

Syntax:

```
implode($delimiter ,  $array );
```

PHP implode Function expects two parameters. One parameter is mandatory while the other is optional. However, it accepts the parameters in either order.

- **$delimiter:** Delimiter or the separator is an optional

  parameter and specifies by what character the array elements will be joined. The default value of delimiter is an empty string.

 $array: $array is an array of strings to implode. It is a mandatory parameter.

Return Value:

PHP implode Function returns a string from the elements of the array separated by the given delimiter.

Ex.

```
implode('-', array('How', 'Are', 'You?');
Output: How-Are-You?

implode(',', array('apple', 'mango', 'banana');
Output: apple,mango,banana
```

2. Explode()

PHP explode Function is an inbuilt string function of PHP. Explode literally means to break down. The Function splits or breaks a string into an array of strings. It returns an array containing exploded sub-strings of the string. Sub-strings form by splitting the string by given string delimiter.

Syntax

```
explode($delimiter, $string);
```

Parameters

PHP Explode Function expects two mandatory parameters and one optional parameter. The description of the parameters is as follows:

 $delimiter: Delimiter or the separator is the mandatory parameter. It specifies where to break the string from. The function will split the string around the delimiter.

 $string: Original string on which explode operation is to be performed. It is also a mandatory parameter.

Return Value

PHP Explode Function returns an array containing the sub-strings of the string exploded. Moreover, the return values also depend upon the limit passed.

```
<?php
$string = "Hi-how-are-you-doing?";
$explodedString = explode('-', $string);
print_r($explodedString);
?>
```

Output

```
Array
(
    [0] => Hi
    [1] => how
    [2] => are
    [3] => you
    [4] => doing?
)
```

3. **Array_flip**()

The PHP array_flip Function is an inbuilt function in PHP which exchanges the keys with their values in the associative array. It returns an array with the corresponding keys and values exchanged. However, the values of the existing array should be valid keys ie. either string or integer. The function throws a warning if the new keys are not valid.
If a value has several occurrences, the latest key will be used as its value, and all others will be lost.
Syntax: array array_flip(array);
Parameters: There is only one parameter in the array_flip Function. The parameter should be an array whose keys and values are to be exchanged.
Return Type:
The PHP array_flip Function returns an array with the keys and values exchanged. However, it returns a NULL value if the input values are invalid.
Ex.

```
<?php
$input = array("oranges", "apples", "pears");
$flipped = array_flip($input);

print_r($flipped);
?>
```

Output:

```
Array
(
    [oranges] => 0
    [apples] => 1
    [pears] => 2
)
```

4. Array_push()
PHP array_push Function is an inbuilt function in PHP which inserts new elements in an array. It always inserts elements at the end of the array. The count of the array is also incremented by one. Moreover, multiple elements can be passed in the array_push function at once.
Syntax:

```
array_push($array, $val1, $val2, $val3 ....);
```

Parameters:

We can pass multiple parameters in the PHP array_push Function. It depends on the number of elements we want to add at once. Let’s consider two types of categories of elements that can be passed:

- **$array:** The first parameter is a mandatory array. The PHP array_push Function inserts new elements in this input array. An empty array can also be passed as the first parameter.
- **$elementList**: The second parameter is the comma-separated element list we want to add to array in PHP.

Return Value:
The function returns the modified array with all the elements inserted at the end.
Note: The PHP array_push method always adds a new numeric key in the original array even if the array has string keys.

Example:

```
<?php
     $testArray = array();
     array_push($testArray, 1);
     print_r($testArray);
      /*
     Array
     (
               [0] => 1
     )
     */
     array_push($testArray, 'Concatly');
     print_r($testArray);
     /*
     Array
     (
               [0] => 1
               [1] => Concatly
     )
     */
     array_push($testArray, array('John', 'Doe'));
     print_r($testArray);
     /*
      Array
     (
               [0] => 1
               [1] => Concatly
               [2] => Array
                    (
                         [0] => John
                         [1] => Doe
                    )
      )
      */
?>
```

5. **array_pop()**

   PHP array_pop Function is an inbuilt function in PHP which removes the last element from an array. It returns the last value of the array. Also, it decrements the size of the array by one. PHP array_pop Function works opposite to array_push function.

   Syntax:

```
array_pop ( array $array );
```

   Parameters:

   PHP array_pop Function takes in only one parameter. The required array from which the element needs to be removed.

   If you don't pass any parameter, PHP array_pop Function throws a Warning. However, it does not throw any warning/notice on passing an empty array.

   Return Value:

   The function returns the last element of the array. The returned element is removed from the array. If the array is empty, then it returns NULL.

   Ex.

```

<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
)

```

6. **count() / sizeof()**

   The count Function in PHP is one of the most used inbuilt functions. It returns the count of elements in an array in PHP. In this article, we will discuss the PHP count Function.

   PHP sizeof Function is an alias of the count function.

   Syntax:

```
      integer count($array, $mode = 0);
```

   Parameters:

   The PHP count Function expects two parameters. One parameter is mandatory while the other is optional. The description of the parameters is given below:

$array: The first parameter is a mandatory array for which you require to count elements. Also, you can pass a multi-dimensional array to it.
$mode: The second parameter to the PHP count is the mode. It is an optional parameter with the default value as 0. It can have two possible values:
0 – Default. Does not count all elements of multi-dimensional arrays.
1 – Counts the array recursively (counts all elements of a multi-dimensional array).

7. sort()

PHP sort is an inbuilt function in PHP. It sorts an array in ascending order i.e, from smaller to bigger element. Also, it makes the changes in the original array itself.

Syntax: boolean sort($array)

Return Value:

The PHP sort Function returns true on success and false on failure. However, it modifies the original input array.

Note: The function assigns new keys to the elements in the array. It will remove any existing keys rather than re-ordering.

Example:

```
<?php
  $array = array(5, 1, 2, 7, 3);
  sort($array);
  print_r($array);
  /*
  Array
  (
      [0] => 1
      [1] => 2
      [2] => 3
      [3] => 5
      [4] => 7
  )
  */
?>
```

8. rsort()

PHP rsort is an inbuilt function in PHP. It sorts an array in descending order i.e, from bigger to smaller element. Also, it makes the changes in the original array itself.

Syntax: boolean rsort($array)

Return Value:

The PHP rsort Function returns true on success and false on failure. However, it modifies the original input array.

Note: The function assigns new keys to the elements in the array. It will remove any existing keys rather than re-ordering.

Example:

```
<?php
      $array = array(5, 1, 2, 7, 3);
      rsort($array);
      print_r($array);
      /*
      Array
      (
          [0] => 7
          [1] => 5
          [2] => 3
          [3] => 2
          [4] => 1
      )
      */
?>
```

9. **asort()**

PHP asort is an inbuilt Function in PHP. It sorts arrays in ascending order while maintaining the relationship between keys and values. The smallest element in the input array appears first and the largest appears last.

Note: This function is mostly helpful in sorting Associative Arrays because the because it preserves the relation between keys and values.

Syntax:
bool asort($array)
Example:

```
<?php

      $array = array('b' => 'banana', 'c' => 'cat', 'a' =>
'apple');
      asort($array);
      print_r($array);
      /*
      Array
      (
          [a] => apple
          [b] => banana
          [c] => cat
      )
      */

?>
```

10. **arsort()**

PHP arsort is an inbuilt Function in PHP. It sorts arrays in descending order while maintaining the relationship between keys and values. The smallest element in the input array appears first and the largest appears last.

Note: This function is mostly helpful in sorting Associative Arrays because the because it preserves the relation between keys and values.

Syntax:
bool arsort($array)
Example:

```
<?php
      $array = array(5, 1, 2, 7, 3);
      arsort($array);
      print_r($array);
         //Keys are preserved
      /*
      Array
      (
          [3] => 7
          [0] => 5
          [4] => 3
          [2] => 2
          [1] => 1
      )
      */

?>
```

11. ksort()

PHP ksort is an inbuilt Function in PHP. It sorts arrays in ascending order according to keys while maintaining the relationship between keys and values. The smallest key in the input array appears first and the largest appears last.

Syntax: bool ksort($array)
Example:

```
<?php
      $testArray = array(1 => 5, 0 => 6, 4 => 3, 3 => 2, 2 =>
1);
      ksort($testArray);
      print_r($testArray);
      /*
      Array
      (
          [0] => 6
          [1] => 5
          [2] => 1
          [3] => 2
          [4] => 3
      )
      */
?>
```

12. krsort()

PHP krsort is an inbuilt Function in PHP. It sorts arrays in descending order according to keys while maintaining the relationship between keys and values. The smallest key in the input array appears last and the largest appears first.

Syntax: bool krsort($array)
Ex.

```
<?php

     $testArray = array(1 => 5, 0 => 6, 4 => 3, 3 => 2, 2 =>
1);
     krsort($testArray);
     print_r($testArray);
     /*
     Array
     (
         [4] => 3
         [3] => 2
         [2] => 1
         [1] => 5
         [0] => 6
     )
     */

?>
```

13. in_array()

The in_array Function is an inbuilt Function in PHP. It is used to check whether an element exists in an array or not. It is one of the most used functions in PHP.

Syntax:
bool in_array ($needle, $haystack, $strictMode = false)

Parameters:
The PHP in_array Function takes in three parameters. Two parameters are mandatory and one is optional. Let's discuss about the parameters below:

$needle: The needle is the first parameter function to the function. It is a mandatory parameter and specifies the element to be searched in the given array. Also, the needle can be an integer, string and even array.
$haystack: The haystack is also a mandatory parameter. It is the array in which to search.
$strictMode: The third parameter to the function is an optional parameter. It specifies whether the search will be strict or not. If the parameter is true then the function matches the value and type of the elements. However, the default value of the parameter is false.

Return Value:

The PHP in_array Function returns a boolean value. If the needle is present in the haystack, then it returns true. Otherwise, it returns false.

Example:

```
<?php
     $haystack = array(5, 0, 6, 21, 63, 24);
     var_dump(in_array(6, $haystack));
     /* bool(true) */

     var_dump(in_array(1, $haystack));
     /* bool(false) */
?>
```

14. array_merge()

PHP array_merge Function is an inbuilt Function in PHP which merges two or more arrays. This function merges elements in two or more arrays into one single array. While merging, it appends the elements of an array at the end of the previous array.

Syntax:

array array_merge($array1, $array2, ……, $arrayn)

Parameters:

The PHP array_merge Function takes in a list of array parameters. You can pass any number of parameters in the function. However, all the parameters must be arrays.

If the input arrays contain the same numeric keys, then the PHP array_merge will append the keys and not overwrite them. However, if the arrays contain the same string keys then it overwrites the later value of the keys with the previous one.

Return Value:

The PHP array_merge Function in PHP returns a single array after merging the arrays passed in the parameters.

Values in the input arrays with numeric keys will be renumbered with incrementing keys starting from zero in the result array

Example:

```
<?php

     $array1 = array(1,3,5);
     $array2 = array(2,4,6);
     $mergedArray = array_merge($array1, $array2);
     print_r($mergedArray);
     /*
     OUTPUT
     Array
     (
          [0] => 1
          [1] => 3
          [2] => 5
          [3] => 2
          [4] => 4
          [5] => 6
     )
     */
```

```
?>
```

Functions

A function is a reusable block of statements that performs a specific task. This block is defined with function keyword and is given a name that starts with an alphabet or underscore. This function may be called from anywhere within the program any number of times

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

According to author of functions there are two types of functions:

1. Built in functions
2. User define functions

User define functions

Function whose definition is defined by user is called as user define function.

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

```
function functionName() {
  code to be executed;
}
functionName(); // function call
```

Rules for function name:

Function name must start with letter or underscore followed by any number, letter or underscore(_)

Function name cannot start with number or any other special symbol

Function names are not case sensitive.

NOTE: You must give function name that shows the working of function

Built in Functions

Built in functions are functions who are already provided by PHP. The real power of php comes from its functions. PHP has more than 1000 functions.

Ex. Echo(), print(), strlen(), print_r()

User define or built in functions may have parameters

According to parameters there are again two types of functions

1. **Parameterized functions**

   Functions who require parameters are called as parameterized function

   Ex.

```
Function welcome($user)
{
      Echo “welcome ”.$user;
}
Welcome(‘Prasad’);
```

According to argument passed there are two types of functions

- **Pass by Value:** On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged. That means a duplicate of the original value is passed as an argument.
- **Pass by Reference:** On passing arguments as pass by reference, the original value is passed. Therefore, the original value gets altered. In pass by reference we actually pass the address of the value, where it is stored using ampersand sign (&).
  - **Ex.**

    ```
    Function add5(&$no)
    {
          $no +=5;
    }
    $n=10;
    Echo “Before function call = $n”
    Add5($n);
    Echo “After function call = $n”;
    // output
    // Before function call = 10
    // After function call = 15
    ```

2. **Non parameterized functions**

   Non parameterized functions don’t require any parameter (arguments)

   Ex.

   ```
   Function welcome()
   {
         Echo ‘welcome user’;
   }
   Welcome();
   ```

Functions may or may not return some value. One function can have one or more return statement but only one statement will be executed at a function call.

Functions with default values

PHP allows us to set default argument values for function parameters. If we do not pass any argument for a parameter with default value then PHP will use the default set value for this parameter in the function call.

Ex.

```php
<?php
function hello($name, $num=12)
{
      echo "$name is $num years old \n";
}

// Calling the function
hello("Ram", 15);

// In this call, the default value 12
// will be considered
hello("Adam");
?>
```

In the above example, the parameter $num has a default value 12, if we do not pass any value for this parameter in a function call then this default value 12 will be considered. Also the parameter $name has no default value , so it is compulsory.

Variable functions

If name of a variable has parentheses (with or without parameters in it) in front of it, PHP parser tries to find a function whose name corresponds to value of the variable and executes it. Such a function is called variable function. This feature is useful in implementing callbacks etc.

```php
<?php
function hello(){
   echo "Hello World";
}
$var="Hello";
$var();
?>
```

Anonymous functions

Anonymous function is a function without any user defined name. Such a function is also called closure or lambda function. Sometimes, you may want a function for one time use. Closure is an anonymous function which closes over the environment in which it is defined.

Syntax:

```php
$var=function ($arg1, $arg2) { return $val; };
```

- ❖ There is no function name between the function keyword and the opening parenthesis.
- ❖ There is a semicolon after the function definition because anonymous function definitions are expressions
- ❖ Function is assigned to a variable, and called later using the variable's name.
- ❖ When passed to another function that can then call it later, it is known as a callback.

Some built in functions

1. **strlen()**

The strlen() is a built-in function in PHP which returns the length of a given string. It takes a string as a parameter and returns its length. It calculates the length of the string including all the whitespaces and special characters.
Syntax:

```
strlen($string);
```

Parameters: The strlen() function accepts only one parameter $string which is mandatory. This parameter represents the string whose length is needed to be returned.

Return Value: The function returns the length of the $string including all the whitespaces and special characters.

Example 1: The below example demonstrates the use of the strlen() function in PHP.

```
<?php

    // PHP program to find the
    // length of a given string
    $str = "GeeksforGeeks";

    // prints the length of the string
    // including the space
    echo strlen($str);
?>
```

Output:

```
13
```

2. **str_word_count()**

The str_word_count() function is a built-in function in PHP and is used to return information about words used in a string like total number word in the string, positions of the words in the string etc.

Syntax:
str_word_count ($string , $returnVal, $chars)

Parameters Used:

$string: This parameter specifies the string whose words the user intends to count.This is not an optional parameter.

$returnVal: The return value of str_word_count() function is specified by the $returnVal parameter. It is an optional parameter and its default value is 0.
The parameter can take below values as required:
0 :It returns the number of words in the string $string.
1 :It returns an array containing all of the words which are found in the string.
2 :It returns an associative array with key-value pairs, where the key deifnes the position of the word in the string and the value is the word itself.

$chars: This is an optional parameter which specifies a list of additional characters which shall be considered as a 'word'.

Return Type: The return type of function depends on the parameter $returnVal and return the values as described above.

Below programs explain the working of str_word_count() function:

```
<?php
$mystring = "Twinkle twinkl4e little star";
print_r(str_word_count($mystring));
?>
```

Output:

```
5
```

3. **strrev()**

Reversing a string is one of the most basic string operations and is used very frequently by developers and programmers. PHP comes with a built-in function to reverse strings.

The strrev() function is a built-in function in PHP and is used to reverse a string. This function does not make any change in the original string passed to it as a parameter.
Syntax:

```
string strrev($inpString)
```

Parameter: This function accepts a single parameter $inpString. This parameter is a string and it specifies the string which we want to reverse. If a number is passed to it instead of a string, it will also reverse that number.

Return Value: The strrev() function returns the reversed string or the number. It does not make any change to the original string or number passed in the parameter.

Examples:

```
Input : $string = "geeks"
Output : skeeg

Input : $string = 143
Output : 341
```

4. strops()
This function helps us to find the position of the first occurrence of a string in another string. This returns an integer value of the position of the first occurrence of the string. This function is case-sensitive, which means that it treats upper-case and lower-case characters differently.

Syntax:

```
strpos(original_str, search_str, start_pos);
```

Parameter value: Out of the three parameters specified in the syntax, two are mandatory and one is optional. The three parameters are described below:

original_str: This is a mandatory parameter that refers to the original string in which we need to search the occurrence of the required string.

search_str: This is a mandatory parameter that refers to the string that we need to search.

start_pos: This is an optional parameter that refers to the position of the string from where the search must begin.

Return Type: This function returns an integer value that represents the index of original_str where the string search_str first occurs.

5. **str_replace()**

The str_replace() is a built-in function in PHP and is used to replace all the occurrences of the search string or array of search strings by replacement string or array of replacement strings in the given string or array respectively.

Syntax:

```
str_replace ( $searchVal, $replaceVal, $subjectVal, $count )
```

Parameters: This function accepts 4 parameters out of which 3 are mandatory and 1 is optional. All of these parameters are described below:

$searchVal: This parameter can be of both string and array types. This parameter specifies the string to be searched and replaced.

$replaceVal: This parameter can be of both string and array types. This parameter specifies the string with which we want to replace the $searchVal string.

$subjectVal: This parameter can be of both string and array types. This parameter specifies the string or array of strings which we want to search for $searchVal and replace with $replaceVal.

$count: This parameter is optional and if passed, its value will be set to the total number of replacement operations performed on the string $subjectVal.

Return Value: This function returns a string or an array based on the $subjectVal parameter with replaced values.

6. **ucfirst()**

The ucfirst() function is a built-in function in PHP which takes a string as an argument and returns the string with the first character in Upper Case and all other characters remain unchanged.

Syntax:

```
ucfirst($string)
```

Parameter: The function accepts only one parameter $string which is mandatory. This parameter represents the string whose first character will be changed to uppercase.

Return Value: The function returns the same string only by changing the first character to upper case of the passed argument $string.

7. **uc_words()**
   The ucwords() function is a built-in function in PHP and is used to convert the first character of every word in a string to upper-case.

   Syntax:

```
string ucwords ( $string, $separator )
```

   Parameter: This function accepts two parameters out of which first is compulsory and second is optional. Both of the parameters are explained below:

   $string: This is the input string of which you want to convert the first character of every word to uppercase.
   $separator: This is an optional parameter. This parameter specifies a character which will be used a separator for the words in the input string. For example, if the separator character is '|' and the input string is "Hello|world" then it means that the string contains two words "Hello" and "world".

   Return value: This function returns a string with the first character of every word in uppercase.

8. **strtolower()**
   The strtolower() function is used to convert a string into lowercase. This function takes a string as parameter and converts all the uppercase english alphabets present in the string to lowercase. All other numeric characters or special characters in the string remains unchanged.
   Syntax:

```
string strtolower( $string )
```

   Parameter: The only parameter to this function is a string that is to be converted to lower case.

   Return value: This function returns a string in which all the alphabets are lower case.

9. **strtoupper()**

The strtoupper() function is used to convert a string into uppercase. This function takes a string as parameter and converts all the lowercase english alphabets present in the string to uppercase. All other numeric characters or special characters in the string remains unchanged.

Syntax:

```
string strtoupper ( $string )
```

Parameter: The only parameter to this function is a string that is to be converted to upper case.
Return value: This function returns a string in which all the alphabets are uppercase.

10. strcmp()
Comparing two strings is one of the most commonly used string operation in programming and web development practices. The strcmp() is an inbuilt function in PHP and is used to compare two strings. This function is case-sensitive which points that capital and small cases will be treated differently, during comparison. This function compares two strings and tells us that whether the first string is greater or smaller than the second string or equals to the second string.

Syntax:

```
strcmp($string1, $string2)
```

Parameters: This function accepts two parameters which are described below:

$string1 (mandatory): This parameter refers to the first string to be used in the comparison
$string2 (mandatory): This parameter refers to the second string to be used in the comparison.

Return Values: The function returns a random integer value depending on the condition of match, which is given by:
Returns 0 if the strings are equal.
Returns a negative value (<0), if $string2 is greater than $string1.
Returns a positive value (>0) if $string1 is greater than $string2.

imagecreate() Function

The imagecreate() function is an inbuilt function in PHP which is used to create a new image. This function returns the blank image of given size. In general imagecreatetruecolor() function is used instead of imagecreate() function because imagecreatetruecolor() function creates high quality images.

Syntax:

```
imagecreate( $width, $height )
```

Parameters: This function accepts two parameters as mentioned above and described below:

$width: It is mandatory parameter which is used to specify the image width.
$height: It is mandatory parameter which is used to specify the image height.

Return Value: This function returns an image resource identifier on success, FALSE on errors.

imagecolorallocate() Function

The imagecolorallocate() function is an inbuilt function in PHP which is used to set the color in an image. This function returns a color which is given in RGB format.

Syntax:

```
int imagecolorallocate ( $image, $red, $green, $blue )
```

Parameters: This function accepts four parameters as mentioned above and described below:

$image: It is returned by one of the image creation functions, such as imagecreatetruecolor(). It is used to create size of image.
$red: This parameter is used to set value of red color component.
$green: This parameter is used to set value of green color component.
$blue: This parameter is used to set value of blue color component.

Return Value: This function returns a color identifier on success or FALSE if the color allocation failed.

imagestring() Function

The imagestring() function is an inbuilt function in PHP which is used to draw a string.

Syntax:

```
bool imagestring( $image, $font, $x, $y, $string, $color )
```

Parameters: This function accepts six parameters as mentioned above and described below:

$image: The imagecreate () function is used to create a blank image in a given size.

$font: This parameter is used to set the font size. Inbuilt font in latin2 encoding can be 1, 2, 3, 4, 5

$x: This parameter is used to hold the x-coordinate of the upper left corner.

$y: This parameter is used to hold the y-coordinate of the upper left corner.

$string: This parameter is used to hold the string to be written.

$color: This parameter is used to hold the color of image.

Return Value: This function returns True on success or False on failure.

imagerectangle()

The imagerectangle() function is an inbuilt function in PHP which is used to draw the rectangle.
Syntax:

```
bool imagerectangle( $image, $x1, $y1, $x2, $y2, $color )
```

Parameters: This function accepts six parameters as mentioned above and described below:

$image: It is returned by one of the image creation functions, such as imagecreatetruecolor(). It is used to create size of image.
$x1: This parameter is used to set the upper left x-coordinate.
$y1: This parameter is used to set the upper left y-coordinate.
$x2: This parameter is used to set the bottom right x-coordinate.
$y2: This parameter is used to set the bottom right y-coordinate.
$color: A color identifier created with imagecolorallocate().

Return Value: This function returns True on success or False on failure.
Note: we can use imagefilledrectangle($image, $x1, $y1, $x2, $y2, $color) for filled rectangle

imagepolygon() Function

The imagepolygon() function is an inbuilt function in PHP which is used to draw a polygon. This function returns TRUE on success and returns FALSE otherwise.
Syntax:

```
bool imagepolygon( $image, $points, $num_points, $color )
bool imagefilledpolygon( $image, $points, $num_points, $color
)
```

Parameters: This function accepts four parameters as mentioned above and described below:

$image: The imagecreatetruecolor() function is used to create a blank image in a given size.
$points: This parameter is used to hold the consecutive vertices of polygon.
$num_points: This parameter contains total number of vertices in a polygon. It must be greater then 3, because minimum three vertices required to create a polygon.
$color: This variable contains the filled color identifier. A color identifier created with imagecolorallocate() function.

Return Value: This function returns TRUE on success or FALSE on failure.

Example:

```
<?php

// Create the size of image or blank image
$image = imagecreate(500, 300);

// Set the vertices of polygon
$values = array(
                50, 50, // Point 1 (x, y)
                50, 250, // Point 2 (x, y)
                250, 50, // Point 3 (x, y)
                250, 250 // Point 3 (x, y)
           );
// Set the background color of image
$background_color = imagecolorallocate($image, 0, 153, 0);
```

```
// Fill background with above selected color
imagefill($image, 0, 0, $background_color);

// Allocate a color for the polygon
$image_color = imagecolorallocate($image, 255, 255, 255);

// Function to create image which contains string.
imagestring($image, 5, 180, 100,  "GeeksforGeeks",
$text_color);

// Draw the polygon
imagepolygon($image, $values, 4, $image_color);

// Output the picture to the browser
header('Content-type: image/png');

imagepng($image);
?>
```

Creating PDF using PHP

Download the fpdf library from fpdf.org
Include fpdf.php file in your code
Fpdf class is defined in the library

Create new object of fpdf class like

```
    $pdf=new fpdf();
```

You can pass few parameters to fpdf object

```
    $pdf=new fpdf("orientation","unit", "size");
```

- Orientation: P or Portrait (default) ,L or Landscape
- unit: pt: point, mm: millimeter (default), cm: centimeter, in: inch
- size: A3, A4(default), A5, Letter, Legal or an array containing the width and the height (expressed in the unit given by unit).

addPage() function

```
$pdf->addPage("orientation","size","rotation");
```

Rotation value must be in multiple of 90;

Setfont() function

```
$pdf->SetFont('Arial','B',16);
```

- B=bold
- I=italic
- U=underline

Cell function

```
Cell(float w , float h , string txt , mixed border , int ln ,
string align , boolean fill , mixed link)
```

- W= Width
- H= height
- Text is string to print
- Border: 0 – no border / 1- border / L –left / T-top/ R-right/ B-Bottom
- Ln: Indicates where the current position should go after the call. Possible values are 0: to the right / 1: to the beginning of the next line / 2: below
- Align: L or empty string: left align (default value) / C: center / R: right align
- Fill : 1 - true/ 0 – false
- Link: URL

Output() function

```
string Output([string dest [, string name]])
```

Dest:

- I: send the file inline to the browser. The PDF viewer is used if available. (default)
- D: send to the browser and force a file download with the name given by name.
- F: save to a local file with the name given by name (may include a path).
- S: return the document as a string.

name

- The name of the file. The default value is doc.pdf.

Other PDF functions:

- **$pdf->ln();**
  This function is used to add new line in pdf file
- **$pdf->SetTextColor(255,0,255);**
  This function is used to set text color in pdf file
- **$pdf->Image("info.png", x position, y position, width, height);**
  We can add images in pdf file using image function
- **$pdf->Line(float x1, float y1, float x2, float y2)**
  we can add line using this function
- **$pdf->SetLineWidth(float width)**
  We can set custom width of lines using this function
- **$pdf->SetDrawColor(rgb);**
  We can change color of lines using this function
- **$pdf->pageNo();**
  We can get page no of pdf file using this function.

Object Oriented Programming in PHP

We can imagine our universe made of different objects like the sun, earth, moon etc. Similarly we can imagine our car made of different objects like wheels, steering, gear etc. Same way there is object oriented programming concepts which assume everything as an object and implement a software using different objects.

Object Oriented Concepts

Before we go in detail, let's define important terms related to Object Oriented Programming.

- ➔ **Class** − This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of objects.
- ➔ **Object** − An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instances.
- ➔ **Member Variable** − These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attributes of the object once an object is created.
- ➔ **Member function** − These are the functions defined inside a class and are used to access object data.
- ➔ **Inheritance** − When a class is defined by inheriting the existing function of a parent class then it is called inheritance. Here a child class will inherit all or few member functions and variables of a parent class.
- ➔ **Parent class** − A class that is inherited from another class. This is also called a base class or super class.
- ➔ **Child Class** − A class that inherits from another class. This is also called a subclass or derived class.
- ➔ **Polymorphism** − This is an object oriented concept where the same function can be used for different purposes. For example, the function name will remain the same but it takes a different number of arguments and can do different tasks.
- ➔ **Overloading** − a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementations.
- ➔ **Data Abstraction** − Any representation of data in which the implementation details are hidden (abstracted).
- ➔ **Encapsulation** − refers to a concept where we encapsulate all the data and member functions together to form an object.

➔ **Constructor** – refers to a special type of function which will be called automatically whenever there is an object formation from a class.

➔ **Destructor** – refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope

Class

A class is a template for objects, and an object is an instance of class.

A class is defined by using the `class` keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods goes inside the braces.

Syntax:

```
<?php
      class Fruit {
        // code goes here...
      }
?>
```

Example:

```
<?php
      class Fruit {
              // Properties
              public $name;
              public $color;

              // Methods
              function set_name($name) {
                $this->name = $name;
              }
              function get_name() {
                return $this->name;
              }
      }
?>
```

Objects

Classes are nothing without objects! We can create multiple objects from a class.

Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class is created using the `new` keyword.

Example;

```
<?php
      class Fruit {
              // Properties
              public $name;
```

```
        public $color;

        // Methods
        function set_name($name) {
          $this->name = $name;
        }
        function get_name() {
          return $this->name;
        }
    }

    $apple = new Fruit();
    $banana = new Fruit();
    $apple->set_name('Apple');
    $banana->set_name('Banana');

    echo $apple->get_name();
    echo "<br>";
    echo $banana->get_name();
?>
```
Output:
```
    Apple
    Banana
```

Constructor

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a `__construct()` function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (__)!

Example:

```
<?php
    class Fruit {
        public $name;
        public $color;

        function __construct($name, $color) {
                $this->name = $name;
                $this->color = $color;
        }
        function get_name() {
            return $this->name;
        }
        function get_color() {
            return $this->color;
        }
    }

    $apple = new Fruit("Apple", "red");
    echo $apple->get_name();
```

```
      echo "<br>";
      echo $apple->get_color();
?>
Output:
      Apple
      red
```

Destructor

A destructor is called when the object is destructed or the script is stopped or exited.

f you create a `__destruct()` function, PHP will automatically call this function at the end of the script.

Notice that the destruct function starts with two underscores (__)!

Example:

```
<?php
      class Fruit {
             public $name;
             public $color;
            function __construct($name) {
                  $this->name = $name;
              }
              function __destruct() {
                  echo "The fruit is {$this->name}.";
              }
      }
      $apple = new Fruit("Apple");
?>
Output:
      The fruit is Apple.
```

Access modifiers

In the PHP each and every property of a class in must have one of three visibility levels, known as public, private, and protected.

- **Public**: Public properties can be accessed by any code, whether that code is inside or outside the class. If a property is declared public, its value can be read or changed from anywhere in your script.
- **Private**: Private properties of a class can be accessed only by code inside the class. So if we create a property that's declared private, only methods and objects inside the same class can access its contents.
- **Protected**: Protected class properties are a bit like private properties in that they can't be accessed by code outside the class, but there's one little difference in any class that inherits from the class i.e. base class can also access the properties.

Generally speaking, it's a good idea to avoid creating public properties wherever possible. Instead, it's safer to create private properties, then to create methods that allow code outside the class to access those

properties. This means that we can control exactly how our class's properties are accessed.
Note: If we attempt to access the property outside the class, PHP generates a fatal error.
PHP Access Specifier's feasibility with Class, Sub Class and Outside World :

Class Memmber Access Specifier	Access from own class	Accessible from derived class	Accessible by Object
Private	Yes	No	No
Protected	Yes	Yes	No
Public	Yes	Yes	Yes

Inheritance

Inheritance is the way of extending the existing class functionality in the newly created class. We can also add some additional functionality to the newly created class apart from extending the base class functionalities. When we inherit one class, we say an inherited class is a child class (sub class) and from which we inherit is called the parent class. The parent class is also known as the base class. This is the way that enables the better management of the programming code and code reusability. The idea behind using the inheritance is all about better management of the code and the code reusability. In this topic, we are going to learn about Inheritance in PHP.
The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.
An inherited class is defined by using the extends keyword.
Example:

```
<?php
class Fruit {
       public $name;
       public $color;
       public function __construct($name, $color) {
              $this->name = $name;
              $this->color = $color;
     }
     public function intro() {
              echo "The fruit is {$this->name} and the color is
     {$this->color}.";
     }
}

// Strawberry is inherited from Fruit
class Strawberry extends Fruit {
       public function message() {
            echo "Am I a fruit or a berry? ";
       }
}

$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

Function overloading and Overriding in PHP

Function overloading and overriding is the OOPs feature in PHP. In function overloading, more than one function can have the same method signature but different number of arguments. But in case of function overriding, more than one functions will have same method signature and number of arguments.

Function Overloading: Function overloading contains same function name and that function performs different task according to number of arguments. For example, find the area of certain shapes where radius are given then it should return area of circle if height and width are given then it should give area of rectangle and others. Like other OOP languages function overloading can not be done by native approach. In PHP function overloading is done with the help of magic function __call(). This function takes function name and arguments.

Example:

```
<?php
// PHP program to explain function
// overloading in PHP

// Creating a class of type shape
class shape {
      // __call is magic function which accepts
      // function name and arguments
      function __call($name_of_function, $arguments) {
            // It will match the function name
            if($name_of_function == 'area') {
                  switch (count($arguments)) {
                        // If there is only one argument
                        // area of circle
                        case 1:
                              return 3.14 * $arguments[0];

                        // IF two arguments then area is rectangle;
                        case 2:
                              return $arguments[0]*$arguments[1];
                  }
            }
      }
}

// Declaring a shape type object
```

```
$s = new Shape;

// Function call
echo($s->area(2));
echo "\n";

// calling area method for rectangle
echo ($s->area(4, 2));
?>
```
Output:

```
      6.28
      8
```

Function Overriding: Function overriding is same as other OOPs programming languages. In function overriding, both parent and child classes should have same function name with and number of arguments. It is used to replace parent method in child class. The purpose of overriding is to change the behavior of parent class method. The two methods with the same name and same parameter is called overriding.

Example:

```
<?php
// PHP program to implement
// function overriding

// This is parent class
class P {

    // Function geeks of parent class
    function geeks() {
        echo "Parent";
    }
}

// This is child class
class C extends P {

    // Overriding geeks method
    function geeks() {
        echo "\nChild";
    }
}

// Reference type of parent
$p = new P;

// Reference type of child
$c= new C;
```

```
// print Parent
$p->geeks();

// Print child
$c->geeks();
?>
```

Output:

```
Parent
Child
```

Object Cloning

An object copy is created by using the clone keyword (which calls the object’s __clone() method if possible). An object’s __clone() method cannot be called directly. When an object is cloned, PHP will perform a shallow copy of all of the object’s properties. Any properties that are references to other variables will remain references.

Syntax:

```
$copy_object_name = clone $object_to_be_copied
```

Ex.

```
<?php
// Program to create copy of an object
// Creating class
class GFG {
    public $data1;
    public $data2;
    public $data3;
}
// Creating object
$obj = new GFG();
 // Creating clone or copy of object
$copy = clone $obj;

// Set values of $obj object
$obj->data1 = "Geeks";
$obj->data2 = "for";
$obj->data3 = "Geeks";

// Set values of copied object
$copy->data1 = "Computer ";
$copy->data2 = "science ";
```

```
$copy->data3 = "portal";

// Print values of $obj object
echo "$obj->data1$obj->data2$obj->data3\n";

// Print values of $copy object
echo "$copy->data1$copy->data2$copy->data3\n";
 ?>
```

Introspection

Introspection in PHP offers the useful ability to examine an object's characteristics, such as its name, parent class (if any) properties, classes, interfaces, and methods.
PHP offers a large number of functions that you can use to accomplish the task.
The following are the functions to extract basic information about classes such as their name, the name of their parent class and so on.

In-built functions in PHP Introspection :

- class_exists(): Checks whether a class has been defined.
- get_class(): Returns the class name of an object.
- get parent_class(): Returns the class name of a Return object's parent class.
- is_subclass_of():Checks whether an object has a given parent class.
- get_declared_classes():Returns a list of all declared classes.
- get_class_methods():Returns the names of the class methods.
- get_class_vars(): Returns the default properties of a class
- interface_exists():Checks whether the interface is defined.
- method_exists(): Checks whether an object defines a method.

Serialization in PHP:

Serialization is a technique used by programmers to preserve their working data in a format that can later be restored to its previous form.
Serializing an object means converting it to a byte stream representation that can be stored in a file.
Serialization in PHP is mostly automatic, it requires little extra work from you, beyond calling the serialize() and unserialize() functions.
Serialize():
The serialize() converts a storable representation of a value.
The serialize() function accepts a single parameter which is the data we want to serialize and returns a serialized string

A serialize data means a sequence of bits so that it can be stored in a file, a memory buffer, or transmitted across a network connection link. It is useful for storing or passing PHP values around without losing their type and structure.
Syntax:

```
      serialize(value);
```

unserialize():
unserialize() can use string to recreate the original variable values i.e. converts actual data from serialized data.
Syntax:

```
      unserialize(string);
```

```
<?php
      $a=array('Shivam','Rahul','Vilas');
      $s=serialize($a);
      print_r($s);
      $s1=unserialize($s);
      echo "<br>";
      print_r($s1);
?>
```

Creating and Validating Forms

HTML Forms

HTML Forms are required when you want to collect some data from the site visitor. For example during user registration you would like to collect information such as name, email address, credit card, etc.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML <form> tag is used to create an HTML form and it has following

syntax:

```
<form action="Script URL" method="GET|POST">
      form elements like input, textarea etc.
</form>
```

Form Attributes:

attribute	**Description**
action	Backend script ready to process your passed data.
method	Method to be used to upload data. The most frequently used are GET and POST methods.
target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
enctype	You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are: **multipart/form-data** - This is used when you want to upload binary data in the form of files like image, word file etc.

HTML Form Controls:

There are different types of form controls that you can use to collect data using HTML form: Text Input Controls

- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

☐ **Text Input Controls**

There are three types of text input used on forms:

Single-line text input controls - This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

Ex. <input type="text" name="first_name" />

Attributes:

➢ Name
➢ value
➢ size
➢ maxlength

Password input controls - This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.

ex. <input type="password" name="password" />

Attributes:

➢ Name
➢ value
➢ size
➢ maxlength

Multi-line text input controls - This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

Ex. <textarea name="address" ></textarea>

Attributes:

➢ Name
➢ rows
➢ cols

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox.

ex.

```
<input type="checkbox" name="maths" value="on"> Maths

<input type="checkbox" name="physics" value="on"> Physics
```

Attributes:

- ➢ Name
- ➢ value
- ➢ checked

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio.

Ex.

```
<form>

      <input type="radio" name="subject" value="maths"> Maths

      <input type="radio" name="subject" value="physics"> Physics

</form>
```

Attributes:

- ➢ Name
- ➢ value
- ➢ checked

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

```
<select name="dropdown">

      <option value="Maths" selected>Maths</option>

      <option value="Physics">Physics</option>

</select>
```

Attributes of <select>:

- ➢ Name
- ➢ multiple
- ➢ size

Attributes of <option> tag:

- ➢ value
- ➢ selected

File Upload Box

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to file.

Ex.

```
<input type="file" name="fileupload" accept="image/*" />
```

Attributes:

- ➢ name
- ➢ accepts

Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input> tag by setting its type attribute to button. The type attribute can take the following values:

submit: This creates a button that automatically submits a form.

reset: This creates a button that automatically resets form controls to their initial values.

button: This creates a button that is used to trigger a client-side script when the user clicks that button.

Hidden form controls

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page has be displayed next based on the passed current page.

Ex.

```
<input type="hidden" name="pagename" value="10" />
```

GET vs. POST Methods:

GET	POST
In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
Get request can be bookmarked.	Post request cannot be bookmarked.
Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent.
Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.

WORKING WITH MULTIPLE FORMS

Web page having many forms:
A web page can have multiple forms with different action, method and different input elements.

```
<html>
	<body>
		<form method="post" action="">
			<input type="text" name="produc">
			<input type="submit">
		</form>

		<form method="post" action="">
			<input type="text" name="produc">
			<input type="submit">
		</form>

	</body>

</html>
```

A form having multiple submit buttons

```
<html>
	<body>
		<form method="post" >
			<input type="text" name="produc">
			<input type="submit">
			<input type="text" name="produc">
```

```
            <input          type="submit"          value=”save”
        formaction=”save.php”>
            <input     type="submit"value=”save     and     next”
        formaction=”savenext.php”>
        </form>

    </body>

</html>
```

Web Page Validation

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct. There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

- Empty String

```
if (empty ($_POST["name"])) {
    $errMsg = "Error! You didn't enter the Name.";
      echo $errMsg;
} else {
    $name = $_POST["name"];
}
```

- Validate String

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
     $ErrMsg = "Only alphabets and whitespace are
allowed.";
                echo $ErrMsg;
} else {
          echo $name;
}
```

- Validate Numbers

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
    $ErrMsg = "Only numeric value is allowed.";
    echo $ErrMsg;
} else {
    echo $mobileno;
}
```

- Validate Email

  A valid email must contain @ and . symbols. PHP provides various methods to validate the email address. Here, we will use regular expressions to validate the email address.

```
<?php
$email = 'abc@abc.c';
$pattern = "^[a-z0-9]+@[a-z0-9-]+\.[a-z0-9]+$^";
if (!preg_match ($pattern, $email) ){
     $ErrMsg = "Email is not valid.";
           echo $ErrMsg;
} else {
     echo "Your valid email address is: " .$email;
}

?>
```

- Input length

```
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);

if ( $length < 10 && $length > 10) {
     $ErrMsg = "Mobile must have 10 digits.";
               echo $ErrMsg;
} else {
     echo "Your Mobile number is: " .$mobileno;
}
```

Complete form validation example:

```
<!DOCTYPE html>
<html>
<head>
<style>
.error {color: #FF0001;}
</style>
</head>
<body>

<?php
// define variables to empty values
$nameErr = $emailErr = $mobilenoErr = $genderErr = $agreeErr = "";
$name = $email = $mobileno = $gender = $agree = "";

//Input fields validation
if ($_SERVER["REQUEST_METHOD"] == "POST")
{

    //String Validation
    if (emptyempty($_POST["name"]))
    {
        $nameErr = "Name is required";
    }
    else
    {
        $name = input_data($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z ]*$/", $name))
        {
            $nameErr = "Only alphabets and white space are allowed";
        }
    }

    //Email Validation
    if (emptyempty($_POST["email"]))
    {
        $emailErr = "Email is required";
    }
    else
    {
        $email = input_data($_POST["email"]);
        // check that the e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL))
        {
            $emailErr = "Invalid email format";
        }
    }

    //Number Validation
    if (emptyempty($_POST["mobileno"]))
    {
        $mobilenoErr = "Mobile no is required";
    }
    else
    {
        $mobileno = input_data($_POST["mobileno"]);
        // check if mobile no is well-formed
        if (!preg_match("/^[0-9]*$/", $mobileno))
        {
            $mobilenoErr = "Only numeric value is allowed.";
        }
        //check mobile no length should not be less and greator than
10
        if (strlen($mobileno) != 10)
        {
            $mobilenoErr = "Mobile no must contain 10 digits.";
        }
    }
    //Empty Field Validation
    if (emptyempty($_POST["gender"]))
    {
        $genderErr = "Gender is required";
    }
    else
    {
        $gender = input_data($_POST["gender"]);
    }

    //Checkbox Validation
    if (!isset($_POST['agree']))
    {
        $agreeErr = "Accept terms of services before submit.";
    }
    else
    {
        $agree = input_data($_POST["agree"]);
    }
}
function input_data($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>

<h2>Registration Form</h2>
<span class = "error">* required field </span>
<br><br>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" >
    Name:
    <input type="text" name="name">
    <span class="error">* <?php echo $nameErr; ?> </span>
    <br><br>
    E-mail:
    <input type="text" name="email">
    <span class="error">* <?php echo $emailErr; ?> </span>
    <br><br>
    Mobile No:
    <input type="text" name="mobileno">
    <span class="error">* <?php echo $mobilenoErr; ?> </span>
    <br><br>
    Website:
    <input type="text" name="website">
    <span class="error"><?php echo $websiteErr; ?> </span>
    <br><br>
    Gender:
    <input type="radio" name="gender" value="male"> Male
    <input type="radio" name="gender" value="female"> Female
    <input type="radio" name="gender" value="other"> Other
    <span class="error">* <?php echo $genderErr; ?> </span>
    <br><br>
    Agree to Terms of Service:
    <input type="checkbox" name="agree">
    <span class="error">* <?php echo $agreeErr; ?> </span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
    <br><br>
</form>

<?php
if (isset($_POST['submit']))
{
    if ($nameErr == "" && $emailErr == "" && $mobilenoErr == "" &&
$genderErr == "" && $websiteErr == "" && $agreeErr == "")
    {
        echo "<h3 color = #FF0001> <b>You have sucessfully
registered.</b> </h3>";
        echo "<h2>Your Input:</h2>";
        echo "Name: " . $name;
        echo "<br>";
        echo "Email: " . $email;
        echo "<br>";
        echo "Mobile No: " . $mobileno;
        echo "<br>";
        echo "Website: " . $website;
        echo "<br>";
        echo "Gender: " . $gender;
    }
    else
    {
        echo "<h3> <b>You didn't filled up the form correctly.</b>
</h3>";
    }
}
?>
```

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

PHP COOKIE

PHP cookie is a small piece of information which is stored in a client browser. It is used to recognize the user.
Cookie is created at server side and saved to client browser. Each time when a client sends a request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax:

```
bool setcookie ( string $name , string $value , int $expire = 0 )
```

Example:

```
setcookie("CookieName", "CookieValue");/* defining name and value
only*/
setcookie("CookieName", "CookieValue", time()+1*60*60);//using
expiry in 1 hour(1*60*60 seconds or 3600 seconds)
```

Access COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.
Ex. $value=$_COOKIE["CookieName"];//returns cookie value

```
<?php
setcookie("user", "demo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
    echo "Sorry, cookie is not found!";
} else {
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Delete Cookie

If you set the expiration date in past, cookie will be deleted.

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

Advantages of Cookies :

1. **Simple to Implement:** Cookies are easy to implement. The fact that cookies are supported on the client’s side means they are a lot easier to implement.
2. **Occupies Less Memory:** Cookies do not require any server resources and are stored on the user’s computer so no extra burden on the server.
3. **Domain Specific:** Each domain has its own cookies. There is no domain that shares cookies with other domains. This makes them independent.
4. **Simple to Use:** Cookies are much simple and easier to use. This is the reason why they are enabled and disabled from the client’s side.
5. **Fast Speed:** The cookies make browsing the Internet faster.

Disadvantages of Cookies:

1. **Not Secured:** As mentioned previously, cookies are not secure as they are stored in the clear text they may pose a possible security risk as anyone can open and tamper with cookies.
2. **Difficult to Decrypt:** We can manually encrypt and decrypt cookies, but it requires extra coding and can affect application performance because of the time that is required for encryption and decryption.
3. **Limitations in Size:** Several limitations exist on the size of the cookie text (4kb in general), the number of cookies (20 per site in general). Each site can hold only twenty cookies.
4. **Can be Disabled:** User has the option of disabling cookies on his computer from the browser’s setting. This means that the user can decide not to use cookies on his browser and it will still work.
5. **Users can Delete Cookies:** The fact that users can delete cookies from their computers gives them more control over the cookies

SESSION

- A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.

Starting a PHP Session: The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP session_start() function is used to begin a new session.It als creates a new session ID for the user.
Below is the PHP code to start a new session:

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

Storing Session Data: Session data in key-value pairs using the $_SESSION[] superglobal array.The stored data can be accessed during the lifetime of a session.
Below is the PHP code to store a session with two session variables Rollnumber and Name:

```
<?php
     session_start();
     $_SESSION["Rollnumber"] = "11";
     $_SESSION["Name"] = "Ajay";
?>
```

Accessing Session Data: Data stored in sessions can be easily accessed by firstly calling session_start() and then by passing the corresponding key to the $_SESSION associative array.
The PHP code to access a session data with two session variables Rollnumber and Name is shown below:

```
<?php
session_start();
echo 'The Name of the student is :' . $_SESSION["Name"] . '<br>';
echo 'The Roll number of the student is :' . $_SESSION["Rollnumber"]
. '<br>';

?>
```

Destroying Certain Session Data: To delete only a certain session data,the unset feature can be used with the corresponding session variable in the $_SESSION associative array.
The PHP code to unset only the "Rollnumber" session variable from the associative session array:

```
<?php
     session_start();
     if(isset($_SESSION["Name"])){
          unset($_SESSION["Rollnumber"]);
     }
  ?>
```

Destroying Complete Session: The session_destroy() function is used to completely destroy a session. The session_destroy() function does not require any argument.

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

Notes:

- The session IDs are randomly generated by the PHP engine .
- The session data is stored on the server therefore it doesn't have to be sent with every browser request.
- The session_start() function needs to be called at the beginning of the page, before any output is generated by the script in the browser.

SENDING E-MAIL

PHP is a server side scripting language that is enriched with various utilities required. Mailing is one of the server side utilities that is required in most of the web servers today. Mailing is used for advertisement, account recovery, subscription etc.
In order to send mails in PHP, one can use the mail() method.
Syntax:
bool mail(to , subject , message , additional_headers , additional_parameters)
Parameters: The function has two required parameters and one optional parameter as described below:

- **to:** Specifies the email id of the recipient(s). Multiple email ids can be passed using commas
- **subject:** Specifies the subject of the mail.
- **message:** Specifies the message to be sent.
- **additional-headers(Optional):** This is an optional parameter that can create multiple header elements such as From (Specifies the sender), CC (Specifies the CC/Carbon Copy

recipients), BCC (Specifies the BCC/Blind Carbon Copy Recipients. Note: In order to add multiple header parameters one must use '\r\n'.

- **additional-parameters(Optional):** This is another optional parameter and can be passed as an extension to the additional headers. This can specify a set of flags that are used as the sendmail_path configuration settings.
- **Return Type:** This method returns TRUE if mail was sent successfully and FALSE on Failure.

Example:

Sending a Simple Mail in PHP

```
<?php
     $to = "recipient@example.com";
     $sub = "Generic Mail";
     $msg="Hello Geek! This is a generic email.";
     if (mail($to,$sub,$msg))
          echo "Your Mail is sent successfully.";
     else
          echo "Your Mail is not sent. Try Again.";
?>
```

Sending a Mail with Additional Options

```
<?php
     $to = "recipient@example.com";
     $sub = "Generic Mail";
     $msg = "Hello Geek! This is a generic email.";
     $headers = 'From: sender@example.com' . "\r\n" .'CC:
     another@example.com';
     if(mail($to,$sub,$msg,$headers))
          echo "Your Mail is sent successfully.";
     else
          echo "Your Mail is not sent. Try Again.";
?>
```

Database Operations

What is MySQL?

MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation.
The data in a MySQL database are stored in tables which consists of columns and rows.

- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, and easy to use database system.It uses standard SQL
- MySQL compiles on a number of platforms.

Connecting to MySQL database using PHP

The mysqli_connect() function in PHP is used to connect you to the database. In the previous version of the connection mysql_connect() was used for connection and then there comes mysqli_connect() where i means improved version of connection and is more secure than mysql_connect().

mysqli_connect ("host", "username", "password", "database_name")

Parameters used:

- **host:** It is optional and it specifies the host name or IP address. In case of local server localhost is used as a general keyword to connect local server and run the program.
- **username:** It is optional and it specify mysql username. In local server username is root.
- **Password:** It is optional and it specify mysql password.
- **database_name:** It is database name where operation perform on data. It also optional.

Return values:

It returns an object which represents a MySql connection. If the connection failed then it return FALSE.

```
<?php
      $servername = "localhost";
      $username = "username";
      $password = "password";
      $db='php_diploma';
      $conn = mysqli_connect($servername, $username, $password,$db);
      // Checking connection
      if (!$conn) {
          die("Connection failed: " . mysqli_connect_error());
      }
      echo "Connected successfully";
?>
```

To close the connection in the mysql database we use the php function mysqli_close() which disconnects from the database. It requires a parameter which is a connection returned by the mysql_connect function.

```
mysqli_close(conn);
```

If the parameter is not specified in mysqli_close() function, then the last opened database is closed. This function returns true if it closes the connection successfully otherwise it returns false.

Insert Data Into MySQL Using MySQLi

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)

VALUES (value1, value2, value3,...)
```

Example:

```
<?php

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "php_diploma";

// Create connection
$link = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$link) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO users (name, mobile_no, email)
VALUES ('abc', '1234567890', 'acb@example.com')";

if (mysqli_query($link, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($link);
```

```
?>
```

Get ID of The Last Inserted Record

After inserting data in mysql table we often need the ID (primary key) of the last inserted record.

mysqli_insert_id() function returns the ID of the last inserted record.

Syntax:

```
$id=mysqli_insert_id($conn);
```

Retrieving the Query Result

mysqli_query() function is used to execute. Data can be fetched from MySQL tables by executing SQL SELECT statements through PHP function mysqli_query().

There are some MySQLi functions used to access data fetched with select query.

1. mysqli_num_rows(mysqli_result $result) which returns a number of rows.
2. mysqli_fetch_row($result) returns a row as a numeric array. it returns null if there are no more rows.
3. mysqli_fetch_assoc(mysqli_result $result) which returns a row as an associative array. Each key of the array represents the column name of the table. It returns NULL if there are no more rows.
4. mysqli_fetch_array($result) returns a row with the combination of numeric and associative array.

Example:

```
<?php
	$servername = "localhost";
	$username = "username";
	$password = "password";
	$dbname = "dbname";

	// Create connection
	$conn = mysqli_connect($servername, $username, $password,
	$dbname);
	// Check connection
	if (!$conn) {
		die("Connection failed: " . mysqli_connect_error());
	}
```

```
     $sql = "SELECT id, first_name, last_name FROM  users";
     $result = mysqli_query($conn, $sql);

     if (mysqli_num_rows($result) > 0) {
       // output data of each row
            while($row = mysqli_fetch_assoc($result)) {
                   echo "id: " . $row["id"]. " - Name: " .
     $row["first_name"]. " " .
                       $row["last_name"]. "<br>";
            }
     } else {
          echo "0 results";
     }

     mysqli_close($conn);
?>
```

UPDATE AND DELETE OPERATIONS ON TABLE DATA

Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function mysql_query.

Update Query:

```
     UPDATE table_name

     SET column1=value, column2=value2,...

     WHERE some_column=some_value
```

Example:

```
<?php
     $servername = "localhost";
     $username = "username";
     $password = "password";
     $dbname = "dbname";

     // Create connection
     $conn = mysqli_connect($servername, $username, $password,
     $dbname);
     $q = "UPDATE usersSET last_name='xyz' WHERE id=2";
     $res=mysqli_query($conn, $sql)
     if (mysqli_affected_rows($res)>0) {
          echo "Record updated successfully";
     } else {
          echo "Error updating record: " . mysqli_error($conn);
     }
     mysqli_close($conn);
```

```
?>
```

Delete records from mysql with PHP

Data can be deleted into MySQL tables by executing SQL DELETE statement through PHP function mysql_query.

Update Query:

```
      DELETE from  table_name

      WHERE some_column=some_value
```

Example:

```
<?php
      $servername = "localhost";
      $username = "username";
      $password = "password";
      $dbname = "dbname";

      // Create connection
      $conn = mysqli_connect($servername, $username, $password,
      $dbname);
      $q = "UPDATE  from users WHERE id=2";
      $res=mysqli_query($conn, $sql)
      if (mysqli_affected_rows($res)>0) {
            echo "Record deleted successfully";
      } else {
            echo "Error deleting record: " . mysqli_error($conn);
      }
      mysqli_close($conn);
?>
```