Unit 1

Web development is all about communication and data instance. This communication is done between two parties client and server. Control by http protocol.

Server :- The server is responsible for providing the web pages depending on the clients requirements it can be either static or dynamic.

Client :- The client is a party that request pages from server and displays them to the end users.

⇒ In general client program is web browser or any other mobile application.

- The user open his web browser.
- The user starts browsing.
- The client forward this request to the server.
- The server acknowledge the request and replies back to the client data.
- The client recieve the page source and the display.
- The client then submit data to the server.
- The server process the data and replies back with a related search result.
- The client again display it back for the user.

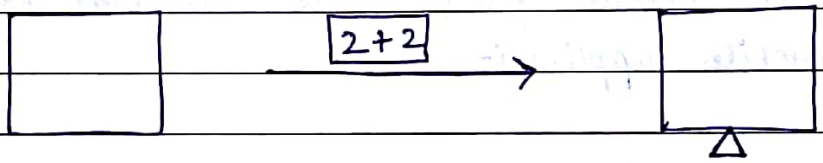
Client side programming :-

In client side programming the entire program run on the client or we can say client side programming deal

with user interface with the user interact in the web. Generally, a browser of the user machine runs the code. in we are code a web page using a client side language and we include a math statement like (2+2) then server will not calculate any thing. The server will simply send the whole code to the client computer and the client computer read the code perform the operation and display no. 4 on the server.

The server send the code 2+2 to the client

The client read the code perform the operations and display the result.



Server Client.

Uses of client side programming :-

- Design and make interaction web pages.
- Interact with temporary storage.
- Works as a interface between user and server.
- Send request to the server retrive data from server.
- Client side programming language

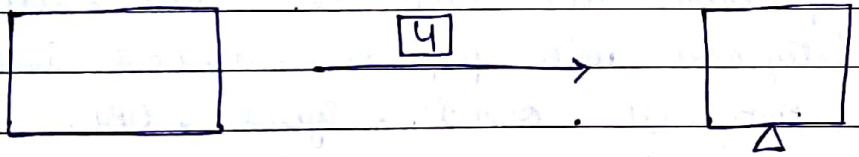
⇒ Javascript, HTML, VB script, CSS, AJAX, J-Query are some client side programming languages.

Server side programming :-

- The type of program directly run on the server and deal with dynamic contents most web pages are not static since they deal with searching database server side language do all the work inside the server before the result is send to the client computer. So if we include 2+2 statement in server side language coded page the server will perform the operation and send '4' to the client computer.

The server read the code 2+2 perform the operation and result to the client.

The client recieve the result and display it.



Uses of server side programming :-

- It process that user input.
- Displays the requested page.
- Interection with servers / storage.
- Interaction with database.
- Execute the Query.
- Some server side programming language.

⇒ PHP, Asp.net, Java, C#, JSA, Python, Ruby.

PHP overview :-

It is a programming languages that allows web developer to create dynamic contents that interact with database. PHP is one of many popular web server side, open source scripting language PHP collect data from pages where user provide and then process to create a dynamic home page output.

- PHP open source and free.
- PHP is object oriented Interpreter languages.
- PHP is server side scripting language that is embedded in HTML.
- PHP web pages works with all major browser.
- PHP is much compact then java and .net.
- PHP is portable and run on linux, windows, plateform.
- It is integrated with popular database including MySQL, PostgreSQL, Oracle, Sybase., DBL.
- PHP support a large no. of network protocol.
- PHP syntax is familar to C, Java.

Php enviroment :- 1. Web server
2. DBMS
3. PHP parser.

PHP program :-
```
<HTML>
<BODY>
<? PHP
 echo "My script is running.";
?>
</BODY>
</HTML>
```

Styles of PHP

1. <? PHP
   —
   —
   —
   ?>

2. SGML
   <?
   —
   —
   —
   ?>

3. ASP style
   <%
   —
   —
   —
   %>

4. HTML Script tag
   < Script language = "PHP">
   —
   —
   —
   </ Script>

⇒ Case Sensitive :-

```
$num = 20;
echo $Num;
```

Data Types in Php :-

1. Integer :- An integer data types is a non-decimal no. between −2147483648 and 2147483647.

2. Double :- A float is a no. with a decimal point or a no. in exponential form.

3. Boolean :- A boolean represents two possible states : TRUE or FALSE.

4. Null :- Null is a special data type which can have only one value : NULL.

5. String :- A string is a sequence of charactors. A string can be any text inside quotes. (single or double quotes.)

6. Array :- An array stores multiple values in one single variable.

7. Object :- An object is a data type which stores data and information on how to process that data.

8. Resources :- A special resources type is not an actual data type. It is the storing of a refrence to function

and resources external to PHP.

Variable :- A variable start with the $ sign, followed by the name of the variable.

Syntax :-

```
$variable name = value;
```

eg :-

```
$ num = 20;
```

Constants :- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

Syntax :-

```
define ("name", "value")
```

eg :-

```
define ("MAX", "50");
echo MAX ;
echo constant ("MAX");
```

Operators :- Operators are used to perform operations on variable and values.

1. Arithmatic operators :- It are used with numeric values to perform common arithmetical operations.

(a) + Addition
(b) - Subtraction

(c) * Multiplication
(d) / Division
(e) % Modulus
(f) ** Exponentiation

2. Assignment operators :- It are used with numeric values to write a value to a variable.

(a) = Assign value
(b) += Addition
(c) -= Subtraction
(d) *= Multiplication
(e) /= Division
(f) %= Modulus.

3. Comparision operators :- It are used to compare two values (number or strings).

(a) == Equal
(b) === Identical
(c) != not equal
(d) <> not equal
(e) !== Not identical
(f) > greater than
(g) < less than
(h) >= greater than or equal to
(i) <= less than or equal to.

4. Increment / Decrement operators :- It are used to increment / decrement a variable's value.

(a) ++ variable — pre increment
(b) variable ++ — post increment
(c) -- variable — pre decrement
(d) variable -- — post decrement

5. Logical Operators :- It are used to combine conditional statement.

(a) and — And
(b) or — Or
(c) xor — Xor
(d) && — And
(e) !! — Or
(f) ! — Not

ARRAY :- An array is a data structure that stores one or more similar type of value in a single value
eg :- If we want to store 100 no. then instead of defining 100 variable and is easy to define an array of 100 length. There are 3 different kinds of an array in php.

1. Indened array
2. Associative array
3. Multidimentional array.

1. Indened Array :- These array can store no. of strings objects but there index will be presented by no. by default array index start from 0.

eg:- ① <HTML>
<BODY>
<? Php
$name = array (16, 19, 17, 15, 19);
for each ($name as $n)
{
echo $n. </br>;
}
?>
</BODY>
</HTML>

② <HTML>
<BODY>
<? Php
$name [0] = "Rohit";
$name [1] = "Mohit";
$name [2] = "Vivek";
for each ($name as $n)
{
echo $n. </br>;
}
?>
</BODY>
</HTML>

③ <HTML>
<BODY>
<? php
$name [0] = "Rohit";
$name [1] = "Mohit";
$name [ ] = "Vivek";

```
foreach ($name as $n)
{
    echo $n . </br>
}
?>
</BODY>
</HTML>
```

④
```
<HTML>
<BODY>
<? php
    $name [] = "Rohit";
    $name [] = "Mohit";
    $name [] = "Vivek";
? foreach ($name as $n)
{
    echo $n . </br>;
}
?>
</BODY>
</HTML>
```

⑤
```
<HTML>
<BODY>
<? Php
    $name [0] = "Rohit";
    $name [4] = "Mohit";
    $name [.] = "Vivek";
foreach ($name as $n)
{
    echo $n . </br>
}
</BODY>
</HTML>
```

2. Associative array :- The associative array are similar to indexed array in term of functionally but they are different in term of there index. Associative array having index as string so that we can establish strong association between key and value. To store the salary of employees in an array a numerically index would not be the best choice instead we could use the employee name as that key in our associative array.

eg :- ①
```
<HTML>
<BODY>
<? php
$ Salary = array ("Rohit" => 4000, "Mohit" => 3000,
                  "Vivek" => 4500);
echo "Salary of Rohit is : ". $ Salary ("Rohit");
echo " Salary of Mohit is :". $ salary ("Mohit");
echo " Salary of Vivek is :". $ salary ("Vivek");
?>
</BODY>
</HTML>
```

②
```
<HTML>
<BODY>
<? PHP
$ salary = array ("Rohit" => 4000, "Mohit" => 3000,
                  "Vivek" => 4500);
for each ($ salary as $n)
{
  echo $ n;
}
??
</BODY>
</HTML>
```

X

Functions of Array:-

1. range() → Creates an array containing a range of elements.

Syntax:-

```
range(low_value, high_value, step);
```

eg:-

```
$a = array(2,9);
$a = array(9,2);
$a = array('a','k');
```

2. count() → Return the no. of elements in an array.

Syntax:-

```
count(arrayname);
```

eg:-

```
$a = array(4,9,16,10);
echo count($a);
```

3. array_slice() → Returns selected parts of an array.

Syntax:- array_slice(arrayname, start_index, last_index).

eg:-

```
$a = array(5,2,9,7,6,5,3,8,7);
$b = array_slice($a, 2,5);
```

4. print_r() → Display array.

Syntax :-

print_r (arrayname)

eg :-

```
$a = array ("Mohit" => 23, "Rohit" => 5);
print_r ($a);
```

Multidimentional Array :-

2D array :- It is a array is array of 1D array.

(i) MD array of index array :-

representation :-

① $Marks = (array (19, 18, 15), array (14, 10, 12), array (20, 18, 17), array (17, 18, 17));

②
```
$Marks [0] [0] = 19;
$Marks [0] [1] = 18;
$Marks [0] [2] = 15;
$ Marks [1] [0] = 14;
$ Marks [1] [1] = 10;
$ Marks [1] [2] = 12;
$ Marks [2] [0] = 20;
$ Marks [2] [1] = 18;
$ Marks [2] [2] = 17;
$ Marks [3] [0] = 17;
$ Marks [3] [1] = 18;
$ Marks [3] [2] = 17;
```

③ $Marks[0] = array(19,18,15);
$Marks[1] = array(14,10,12);
$Marks[2] = array(20,18,17);
$Marks[3] = array(17,16,17);

2. MD of Associative array :-

representation :-

① $Marks = array("Mohit" ⇒ array("Phy" ⇒ 19, "Che" ⇒ 18, "Math" ⇒ 15),
"rohit" ⇒ array("Phy" ⇒ 14, "che" ⇒ 10, "Math" ⇒ 12),
"vinek" ⇒ array("Phy" ⇒ 20, "che" ⇒ 18, "Math" ⇒ 17),
"Raj" ⇒ array("Phy" ⇒ 17, "che" ⇒ 18, "Math" ⇒ 17));

② $Marks["Mohit"]["Phy"] = 19;
$Marks["Mohit"]["che"] = 18;
$Marks["Mohit"]["Math"] = 15;
$Marks["Rohit"]["Phy"] = 14;
$Marks["Rohit"]["che"] = 10;
$Marks["Rohit"]["Math"] = 12;
$Marks["Vinek"]["Phy"] = 20;
$Marks["Vinek"]["Che"] = 18;
$Marks["Vinek"]["Math"] = 17;
$Marks["Raj"]["Phy"] = 17;
$Marks["Raj"]["che"] = 18;
$Marks["Raj"]["Math"] = 17;

③ $Marks["Mohit"] = array("Phy" ⇒ 19, "che" ⇒ 18, "Math" ⇒ 15);
$Marks["Rohit"] = array("Phy" ⇒ 14, "che" ⇒ 10, "Math" ⇒ 12);
$Marks["Vinek"] = array("Phy" ⇒ 20, "che" ⇒ 18, "Math" ⇒ 17);
$Marks["Raj"] = array("Phy" ⇒ 17, "che" ⇒ 18, "Math" ⇒ 17);

Strings :- A string is a sequence of characters where a character is a same age a byte (1 byte) this means php support only 256 character set.

eg :- ' This is a single quoted string '
" This is a double quoted string "
' R '
" R "
' Subodh '
" College "

Types of string :-

1. Single quote string
2. Double quote string
3. here docs string

1. Single quote string :- The simplest way to specify a string is to enclose it in single quotes. when a string specified with single quote all variables in the string will not be consider as variable.

Syntax :-

' string '

eg :- $num = 10;
echo ' the value of num is $num ';
echo ' the use of \n escape sequence ';

output :- the value of num is $num.
the use of \n escape sequence.

2. Double quotes String :- The simplest way to specify a string is to enclose it in double quotes. Double quoted string replace variables with there values as well as escape sequences.

syntax :- " String "

eg :- $num = 10 ;
echo " the value of num is $num " ;
echo " the use of \n escape sequence " ;

Output :- the value of num is 10 :
the use of
escape sequence.

3. Here docs String :- It is used to embbed a large peice of text in our string which may include lot of single quote double Quote without having to constantly to escape them.

syntax :- <<< docs stringname - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - -

docsstringname ;

eg:- echo <<< START - - - - -
- - - - - - - - - - -
- - - - - - - - - - -

START ;

Escape Sequence :-

1. \' → To escape ' within single quoted string.
2. \" → To escape " within double quoted string.
3. \\ → To escape the backslash.
4. \$ → To escape $.
5. \n → To add line breaks between strings.
6. \t → To add tab space.
7. \r → For carriage return.

String functions :-

1. strlen() :- Returns the length of a strings
   syntax :-
   strlen ("string");
   eg :-
   strlen ("Raj");

2. strpos() :- Returns the position of the first occurrence of a strings inside another strings.
   syntax :-
   strpos (strings, substring);
   eg :-
   echo strpos ("Hello World", "World");

3. strtoupr() :- Converts a strings to uppercase letters.

syntax :-

```
strtoupr (string);
```

eg :-

```
$n = "rAj"
$a = strtoupr ($n);
```

4. strtolwr() :- Converts a strings to lowercase letters.

syntax :-

```
strtolwr (string);
```

eg :-

```
$n = "Raj"
$a = strtolwr ($n);
```

5. substr() :- Returns a part of a strings.

syntax :-

```
substr (string, start, length);
```

eg :-

```
$str1 = "Hello World";
$str2 = substr ($str1, 2);
echo $str2;
```

6. strrev :- Reverse a string.

syntax :-

```
strrev (string);
```

eg :-

```
strrev ("Hello");
```

7. ord() :- Returns the ASCII value of the first character of a string.

syntax:-

ord (string);

eg:-

echo ord ("a");

8. chr ():- Returns a character from a specified ASCII value

syntax:-

chr (ASCII);

eg:-

chr (97);

9. str_split ():- Split the string.

syntax:-

str_split (string, chunksize);

eg:-

10. Explode ():- Break a string into two an array.

syntax:-

explode (separator, string, limit);

eg:-

$s1 = "This is my school";
$result = explode ('s', $s1);

11. substr_count ():- Counts the no. of times a substring occures in a string.

syntax:-

substr_count (string, substring);

eg:-

$str = "Hello world";
echo substr_count ($str, "world");

Class in PHP :- A class is a self-contained, independent collection of variable and function which work together to perform one or more specific tasks, while objects are individual instances of a class.

syntax:-

```
Class classname
{

}
```

eg:-

```
Class demo
{
    function xyz()
    {
        echo "An example of OOPS";
    }
};
```

Object in PHP :- An object is like an actual house built according to that blueprint.

syntax:-

```
classname objectname
```

eg:-

```
demo d1;
$d1 = new demo();
```

. Function calling in PHP :-

The arrow symbol (->) is an OOP construct that is used to access contained properties and methods of a given object.

Syntax :-

```
object -> function_name;
```

eg :-

```
$d1 -> xyz();
```

eg :- ①

```
<HTML>
<BODY>
<?PHP
    class student
    {
      $name;
      function SetName ($st)
      {
        $this -> name = $st;
      }
      function showname()
      {
        echo $this.name;
      }
    };
$rohit = new student();
$rohit -> SetName ("Rohit Kumar");
$rohit -> showname();
?>
</BODY>
</HTML>
```

② <HTML>
<BODY>
<?PHP
class student
{
$rn;
$per;
function SetData ($rn , $per)
{
$this rn = $rn;
$this → per = $per;
}
function showData()
{
echo $this rn." ". $this →per;
}
};
$rohit = new student();
$rohit → SetData (101, 69.48);
$rohit → show_Data();
?>
</BODY>
</HTML>

Visibility of members:-

1. Public
2. Protected
3. Private.

1. Public :- Public method or variable can be accessible from anywhere.

2. Protected :- Method or variable with protected visibility can only be access in the derived class.

3. Private :- Method or property with private visibility can only be accessible inside the class.

Constructor :- Constructor is a special type of method that is used to initialized the object.

(i) Same name
(ii) No value return
(iii) Execute automatically when object are created.
(iv) Initialize object
(v) public access specifier
(vi) parameterised constructor
(vii) constructor overloading
(viii) explicit constructor
(ix) Implicit constructor.

Destructor :-

(i) Same name with tilde symbol (~)
(ii) No value return
(iii) Execute automatically when object are destroyed
(iv) public access specifier
(v) explicit destructor
(vi) implicit destructor.

eg:- ① constructor :-

```
<HTML>
<BODY>
<?PHP
    class student
    {
      $name;
      function student ($n)
      {
        $this -> name = $n;
      }
      function getName()
      {
        echo $this -> name;
      }
    };
    $ob = new student ("Rohit Kumar");
    $ob -> getName();
?>
</BODY>
</HTML>
```

eg:- ② Destructor :-

```
<HTML>
<BODY>
<?PHP
    class student
    {
      var $name;
      function ~ student
      {
        $this -> name = $n;
      }
```

```
function getname()
{
    echo $this -> name;
}
};
$ob = new student ("Rohit Kumar");
$ob -> getname();
?>
</BODY>
</HTML>.
```

Q. How Data send to server?

A. Before the browser send the information, it encode it by using a scheme for url encoding. In this scheme, name/value pairs are joint with (=) sign and different pairs are seprated by (&).

for eg:-

name1 = value1 & name2 = value2 & name3 = value3;

Note:-⇒ Spaces are removed or replaced or replace by (+) sign and any other non-alpha numeric character are replaced with a hexadecimal value after the information is encoded it is send to the web browser.

Form datahandling:-

★ There are two ways the browsers client can send information to the web server:-

1. POST method
2. GET method.

1. POST method :- PHP $_POST is widely used to collect form data after submitting an HTML form with method = "POST".

→ The POST method transfers information via http header.

→ The POST method doesn't have any restriction on data size to be sent.

→ The POST method can be used to send ASCII as well as binary data.

→ The Data sent by POST method goes through http headers so security depends on http protocol.

→ The PHP provide $_POST associative array to access all the information sent by POST method.

eg :- ① B.PHP

```
<HTML>
<BODY>
<form action = "A.php"  method = "POST">
Enter your name : <input type = "text"
                  name = "sname">
<input type = "submit" />
</form>
</BODY>
</HTML>
```

eg:- ② A.php

```
<HTML>
<BODY>
<? PHP
    echo "Good Morning". $_POST["sname"];
?>
</BODY>
</HTML>
```

2. GET Method :- $_GET can also used to collect form data after submitting an HTML form with method = "GET".

→ $_GET can also collect data sent in the url the page and the encoded information are seprated by (?) symbol.

→ http://xyz.com/a.php?sname=Ravi+jain&fname=Rgj+Jain

→ The GET method produce a long string that appears in the browser location box.

→ The GET method is restricted to send upto 2000 characters only.

→ Never use GET method if we have password or other sensitive information to be send to the server.

→ The data send by GET method can be access

using QUERY_STRING enviroment variables.

→ PHP provide $_GET associative array to access all the sent information using.

→ The variables are displayed in the url it is possible bookmark the page.

eg:- ① B.php

```
<HTML>
<BODY>
<form action = "A.php" method = "GET">
Enter Your name : <input type = "text" name = "sname">
<input type = "submit" />
</form>
</BODY>
</HTML>
```

eg:- ② A.php

```
<HTML>
<BODY>
<?PHP
    echo "Good Morning". $_GET["sname"];
?>
</BODY>
</HTML>
```

3. $_REQUEST variable:- It is contains the contents of both $_GET & $_POST and $_COOKIE.

eg:- ①
```
<HTML>
<BODY>
<? PHP
  echo "Good Morning". $_REQUEST ["sname"];
?>
</BODY>
</HTML>
```

COOKIES:- It are small text file that the server create on the user computers and they are kept of tracking purpose. Each time the same computer request a page with a browser, it will send the cookies too. We can create and retrive cookies value by using $_COOKIE variable. Http cookies are parsel of text sent by server to a web client and then sent back unchanged by the client each time it access the server.

eg:- ①
```
<? PHP
  setcookie ("user", "Rohit", time()+3600);
?>
<HTML>
<BODY>
  <? PHP
    if (isset ($_COOKIE ["user"]))
    {
      echo "Welcome". $_COOKIE ["user"];
    }
    else
    {
```

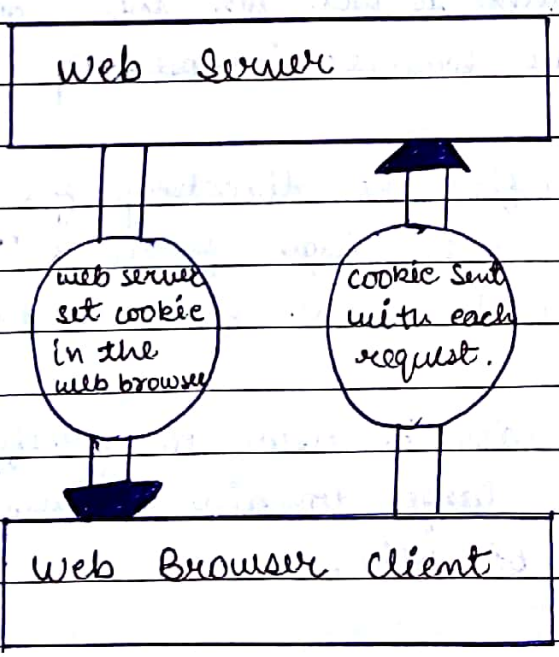
```
    echo "Welcome Guest!";
}
?>
</BODY>
</HTML>
```

Each cookie has a name and value it may also have expiry time. Cookie are send by web server inside the http response before any content. cookie are store in the client computer. There are 3 steps.

(a) Server script send a set of cookie to the browser.

(b) Browser store this information on local machine for future use.

(c) when next time browser send ~~by~~ any request to web server when it send those cookie information to the server use as that information to identify.



How to create cookies :-

PHP provide setcookie method to create a cookie this function requires 6 arguments and should be called before html tag.

Syntax :-
setCookie ("name", "value", expiration time, path, domain, security).

1. Name :- This argument set the name of cookie and it is store in environment variable called http_cookie_vars.

2. Value :- This argument set the value of cookie and is a content that we actually want to store.

3. Expiration time :- This specifies a future time in second after this time cookie will become inaccessible If this parameter is not set then cookie will automatically expire when the browser is close.

4. Path :- This specifies the directory for which the cookie is valid the single forward "/" permits the cookie to be valid for all the directories.

5. Domain :- This can be used to specify the domain name in very large domain & must contain atleast two period to be valid.

6. Security :- This can be set '1' to specify that the

cookie should only be sent by secure transmission using http otherwise set to '0' which mean cookie can be sent by regular http.

Access the cookies:-
The $_COOKIE variable is use to access cookies value and it is a super global variable.

Delete the cookies:-
When Deleting a cookie we should assure that the expiration time is in the pass.

ARRAY FUNCTIONS:-

1. Sort:- Sort Indexed array in assending order.

Syntax:-
```
sort (arrayname);
```

eg:-
```
<?PHP
    $a = array (16, 12, 2, 17, 19)
    sort ($a);
    Print_r ($a);
?>
```

output :- 2 12 16 17 19

2. rsort :- Sort an indexed array in decending order.

Syntax :-
```
rsort (arrayname);
```

eg :-
```
<?PHP
    $a = array (16,12,2,19,17);
    rsort ($a);
    Print_r ($a);
?>
```

output :- 19 17 16 12 2

3. asort :- Sort associative array in accending order according to the value.

Syntax :-
```
asort (arrayname);
```

eg :-
```
<?PHP
    $a = array ("Rohit" => 19, "Mohit" => 17,
                "vivek" => 18);
    asort ($a);
    print_r ($a);
?>
```

Output :- $a[Mohit] = 17
$a[vivek] = 18
$a[Rohit] = 19.

4. Ksort :- Sort Associative array in accending order according to the key.

Syntax:-

```
Ksort (arrayname);
```

eg:-

```
<?PHP
    $a = array ("Rohit"=> 19 , "Mohit"=>17 , "Vivek"=>18);
    Ksort ($a);
    print_r ($a);
?>
```

Output:-

```
$a[Rohit] = 19
$a[Vivek] = 18
$a[Mohit] = 17
```

5. arsort :- Sort Associative array in decending order according to the value.

Syntax:-

```
arsort (arrayname);
```

eg:-

```
<?PHP
    $a = array ("Rohit"=>19, "Mohit"=>17 , "Vivek"=>18);
    arsort ($a);
    print_r ($a);
?>
```

Output :-

$a [Rohit] = 19
$a [vivek] = 18
$a [Mohit] = 17

6. Krsort :- Sort Associative array in decending order according to the key.

Syntax :-

Krsort (arrayname);

eg :-

```
<?PHP
  $a = array ("Rohit" => 19, "Mohit" => 17,
              "vivek" => 18);
  Ksort ($a);
  print_r ($a);
?>
```

Output :-

$a [vivek] = 18
$a [Rohit] = 19
$a [Mohit] = 17,

Session Management :-

Session variable are used to access variable in different pages or Multiple pages of a web application. A session a create a file in an temporary directory on the server where registered session variables and there value are stored. This data will be

available to all pages on the site during that visit. When a session is start following things happens.

i) PHP first create a unique 32 digit hexadecimal code for that particular session like A1687B2A189A13A1.

ii) A cookie named PHPSESSID is automatically sent to the users computers to store unique sessionID string a file is automatically created on the server in the temporary directory having the name of unique ID prefix by

eg:-

SESS_A1687B2A189A13A1.

When a PHP page wants to retrive the value from a session variable PHP automatically get the unique sessionID string from the PHPSESSID cookie and then search in its temporary directory for the files bearing that name. A session ends when the user close the browser or after leaving the site.

eg:- ① a.php.

```
<?PHP
    SESSION_start()
    if (isset ($_SESSION["num"]))
    {
        $_SESSION["num"] = $_SESSION["num"]+1;
    }
    else
        $_SESSION["num"]=1;
?>
```

```
<HTML>
<BODY>
 <?PHP
      echo "you have visited this page".
           $_SESSION["Num"]."times in this
           session";
 ?>
 <br>
  <a href="b.php"> Next </a>
 </BODY>
 </HTML>
```

(2) b.php.

```
<HTML>
 <BODY>
   <?PHP
      echo "Hello";
   ?>
 <br>
 <a href="a.php"> back </a>
  </BODY>
  </HTML>
```

Starting a PHP session :-

A PHP session is started by calling a function SESSION_start() function this function first checks if a session is already started and if non is started then starts one. We should call SESSION_start() function at the beginning of page before HTML tag. SESSION variables are store is an associated array called $_SESSION[] these

variables can be accessed during life time of the session.

Deleting a PHP session :-
A PHP session is destroyed by a method SESSION_destroy() this function doesn't need any arguments and single call can destroy all the session variables if we want to destroy a single session variable then we can use unset method by passing session variable name and argument.

File handling :- File handling is an important part of any web application. You often need to open and process a file for different tasks.

functions in file handling :-

(1) include() :- It will only produce a warning and the script will continue.

Syntax :-

```
include (filename);
```

eg :-

```
<?PHP
    include ("first.php");
?>
```

(2) file_exists():- Checks weather or not a file or directory exists.

Syntax:-

```
file_exists(filename);
```

eg:-

```
<?PHP
if (file_exists ("xyz.php"))
{
    echo "yes";
}
else
{
    echo "No";
}
?>
```

(3) is_file():- Checks weather a file is a regular file.

Syntax:-

```
is_file(filename);
```

eg:-

```
<?PHP
if (is_file ("xyz.php"))
{
    echo "yes";
}
```

```
else
  {
    echo "No";
  }
?>
```

4. is_dir():- Checks whether a file is a directory.

Syntax:-

```
is_dir( filename)
```

eg:-

```
<?PHP
  if(is_dir("xyz.php"))
    {
      echo "yes";
    }
  else
    {
      echo "No";
    }
?>
```

5. is_readable():- Checks whether a file is a directory.

Syntax:-

```
is_readable(filename)
```

eg:-

```
<?PHP
```

```
if (is_readable ("xyz.php"))
  {
    echo "yes";
  }
else
  {
    echo "No";
  }
?>
```

6. is_writable():- Checks whether a file is writeable.

Syntax:-

```
is_writable ( filename)
```

eg:-

```
<? PHP
    if (is_writable ("xyz.php"))
    {
      echo "yes";
    }
    else
    {
      echo "No";
    }
?>
```

7. is_executable():- checks whether a file is executable.

Syntax:-

```
is_executable(filename);
```

eg:-

```
<?PHP
    if (is_executable("xyz.php")):-
    {
        echo "yes";
    }
    else
    {
        echo "No"
    }
?>
```

8. Filesize():- Return the ~~size~~ file size.

Syntax:-

```
Filesize(filename);
```

eg:-

```
<?PHP
    echo filesize("circle.php")
?>
```

9. filectime():- Returns the late change time of a file.

Syntax:-

```
filectime("circle.php");
```

eg:- <?PHP
$a = filectime ("circle.php");
echo date ("D d M Y g:i A", $a);
?>

10. fileatime():- Returns the last access time of a file.

syntax:-
fileatime(filename);

eg:-
<?PHP
$a = fileatime ("circle.php");
echo date ("D d M Y g:i A", $a);
?>

11. filemtime():- Return the last modification time of a file.

syntax:-
filemtime(filename);

eg:-
<?PHP
$a = filemtime ("circle.php");
echo date ("D d M Y g:i A", $a);
?>

12. touch():- sets access and modification time of a file.

Syntax :-

touch (filename);

eg :-

```
<? PHP
    touch ("test.txt");
?>
```