Part - A (Each question carries Two marks)

UNIT – I

1. What is Echo Statement?

Answer:

In PHP the two basic constructs to get outputs are echo and print. Actually, echo() is not a function, it is a language construct, therefore, you can use it without parentheses.

Syntax

```
echo (arg1, arg2... )
```

Example : Simple string display

```
<?php

echo 'One line simple string.<br />';

echo 'Two line simple string example<br />';

echo 'Tomorrow I \'ll learn PHP global variables.<br />';

echo 'This is a bad command : del c:\\*.* <br />';

?>
```

All the above echo commands simply display the corresponding string, here we have used an additional html command
 at the end of each echo statement to generate a line break as \n cannot generate a line break in browser.

Example : Variable inside echo statement

```
<?php
// Variables inside an echo statement.
$abc='We are learning PHP';
$xyz='w3resource.com';
echo "$abc at $xyz <br />";
// Simple variable display
echo $abc;
echo "<br />"; // creating a new line
echo $xyz;
echo "<br />"; // creating a new line
// Displaying arrays
$fruits=array('fruit1'=>'Apple','fruit2'=>'Banana');
echo "Fruits are : {$fruits['fruit1']} and
{$fruits['fruit2']}" ;
?>
```

Output:

```
We are learning PHP at w3resource.com
We are learning PHP
w3resource.com
Fruits are : Apple and Banana
```

2. What is Print Statement?

Answer:

Print is used to display a string.

Syntax

```
print(string $val)
```

Parameters

val : The input data.

Return Values

Returns 1, always.

Print() is not actually a real function, it is a language construct like echo. There is some difference between the two, echo is marginally faster compare to print as echo does not return any value. Echo can accept multiple parameters without parentheses but print can accept only one parameter.

Example to display simple string with print()

```
<?php
print 'One line simple string.<br />';
print 'Two lines simple
string example<br />';
print 'Tomorrow I \'ll learn PHP global variables.<br />';
print 'This is a bad command : del c:\\*.* <br />';
```

```
?>
```

All the above print commands simply display the corresponding string, here we have used an additional html command
 at end of each print statement to generate a line break.

Output:

```
One line simple string.
Two lines simple  string example
Tomorrow I 'll learn PHP global variables.
This is a   bad command : del c:\*.*
```

View the example in the browser

Advance example of PHP print()

```
<?php
// Variables inside an echo statement.
$abc='We are learning PHP';
$xyz='w3resource.com';
print "$abc at $xyz <br />";
// Simple variable display
print $abc;
print "<br />"; // creating a new line
print $xyz;
print "<br />"; // creating a new line
// Displaying arrays
$fruits=array('fruit1'=>'Apple','fruit2'=>'Banana');
print "Fruits are : {$fruits['fruit1']} and {$fruits['fruit2']}" ;
?>
```

Output:

```
  We are learning PHP at w3resource.com
  We are learning PHP
  w3resource.com
  Fruits are : Apple and Banana
```

3. What is HTTP Protocol?

Answer: The HTTP functions let you manipulate information sent to the browser by the Web server, before any other output has been sent.

The HTTP functions are part of the PHP core. There is no installation needed to use these functions.

HTTP Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description
header()	Sends a raw HTTP header to a client
headers_list()	Returns a list of response headers sent (or ready to send)
headers_sent()	Checks if / where the HTTP headers have been sent
setcookie()	Defines a cookie to be sent along with the rest of the HTTP headers
setrawcookie()	Defines a cookie (without URL encoding) to be sent along with the rest of the HTTP headers

4. What is variable in PHP?

Answer:

Variables are "containers" for storing information.

A variable is a container for data. Once data has been stored in a variable (or, stated more accurately, once a variable has been assigned a value), that data can be altered, printed to the Web browser, saved to a database, emailed, and so forth. Variables in PHP are, by their nature, flexible: You can put data into a variable, retrieve that data from it (without affecting the value of the variable), put new data in, and continue this cycle as long as necessary. But variables in PHP are largely temporary: Most only exist—that is, they only have a value—for the duration of the script's execution on the server. Once the execution passes the final closing PHP tag, those variables cease to exist. Furthermore, after users click a link or submit a form, they are taken to a new page that may have an entirely separate set of variables.

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

```
Output:
```

Hello world!
5
10.5

5. What is # statement in PHP?

Answer:

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

Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

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

PHP - The if Statement

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

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

PHP - The 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;
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```
<?php
$t = date("H");

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

PHP - The 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 this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example

```
<?php
$t = date("H");

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

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

The PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

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

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example

```
<?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!";
}
?>
```

6. What is <?php?

Answer:

When PHP parses a file, it looks for opening and closing tags, which are *<?php* and *?>* which tell PHP to start and stop interpreting the code between them. Parsing in this manner allows PHP to be embedded in all sorts of different documents, as everything outside of a pair of opening and closing tags is ignored by the PHP parser.

PHP also allows for short open tag *<?* (which is discouraged since it is only available if enabled using theshort_open_tag `php.ini` configuration file directive, or if PHP was configured with the --enable-short-tagsoption).

If a file is pure PHP code, it is preferable to omit the PHP closing tag at the end of the file. This prevents accidental whitespace or new lines being added after the PHP closing tag, which may cause unwanted effects because PHP will start output buffering when there is no intention from the programmer to send any output at that point in the script.

```
<?php
echo "Hello world";
```

```
// ... more code

echo "Last statement";

// the script ends here with no PHP closing tag
```

7. What is action in form control?

answer: A `form` is useless unless some kind of processing takes place after the form is submitted. The action attribute is used to inform the browser what page (or script) to call once the `"submit"` button is pressed.

Here, the `action` attribute tells the browser to send the `form` data to a form-handling PHP page (which will presumably convert the form data to something more email-friendly):

```
<form action="form-to-email.php" method="post"
    accept-charset="windows-1252">
  <div>
    <label for="txtname">Name:</label>
    <input type="text" name="txtname" id="txtname"/>
  </div>
 ⋮
</form>
```

Value

This element takes as its value a URL to a document that may be on the same server (for example, a shared CGI folder that has various form-processing scripts), or even a page or script on an entirely separate server (perhaps a free form-handling service).

8. What is method in Form control?

answer:

The Method Attribute

The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data:

Example

```
<form action="/action_page.php" method="get">
```

When to Use GET?

The default method when submitting form data is GET.

However, when GET is used, the submitted form data will be **visible in the page address field**:

```
/action_page.php?firstname=Mickey&lastname=Mouse
```

Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user want to bookmark the result
- GET is better for non-secure data, like query strings in Google

When to Use POST?

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

Notes on POST:

- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

9. What is Server Page?

answer:

A server is a computer program that provides a service to another computer programs (and its user). In a data center, the physical computer that a server program runs in is also frequently referred to as a server. That machine may be a dedicated server or it may be used for other purposes as well.

In the client/server programming model, a server program awaits and fulfills requests from client programs, which may be running in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a *server* of requests from other programs.

Types of servers

Servers are often categorized in terms of their purpose. A Web server, for example, is a computer program that serves requested HTML pages or files. The program that is requesting web content is called a client. For example, aWeb browser is a client that requests HTML files from Web servers.

Here are a few other types of servers, among a great number of other possibilities:

An application server is a program in a computer in a distributed network that provides the business logic for an application program.

A proxy server is software that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service.

A mail server is an application that receives incoming e-mail from local users (people within the same domain) and remote senders and forwards outgoing e-mail for delivery.

A virtual server is a program running on a shared server that is configured in such a way that it seems to each user that they have complete control of a server.

A blade server is a server chassis housing multiple thin, modular electronic circuit boards, known as server blades. Each blade is a server in its own right, often dedicated to a single application.

A file server is a computer responsible for the central storage and management of data files so that other computers on the same network can access them.

A policy server is a security component of a policy-based network that provides authorization services and facilitates tracking and control of files.

10. What is Client page?

answer:

A client is the receiving end of a service or the requestor of a service in a client/server model type of system. The client is most often located on another system or computer, which can be accessed via a network. This term was first used for devices that could not run their own programs, and were connected to remote computers that could via a network. These were called dumb terminals and they were served by time-sharing mainframe computers.

A client can be a simple application or a whole system that accesses services being provided by a server. A client can connect to a server through different means like domain sockets, named, shared memory or through Internet protocols, which is the most common method being used since the wide adoption of the Internet.

Clients are classified into three types:

- Thin Client: A client application with minimum functions that uses the resources provided by a host computer and its job is usually just to display results processed by a server. It simply relies on a server to do most or all of its processing.
- Thick/Fat Client: This is the opposite of the thin client. It can do most of its processing and does not necessarily rely on a central server, but may need to connect to one for some information, uploading, or to update data or the program itself. Anti-virus software belong to this category because they do not really need to connect to a server to do their job, although they must connect periodically to download new virus definitions and upload data.
- Hybrid: Exhibits characteristics from the two above types. It can do most processes on its own but may rely on a server for critical data or for storage.

(Each question carries Two marks)

UNIT – II

1. What is Form Control in PHP?

Answer:

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html

```
1. <form action="welcome.php" method="get">
2. Name: <input type="text" name="name"/>
3. <input type="submit" value="visit"/>
4. </form>
```

File: welcome.php

```
1. <?php
2. $name=$_GET["name"];//receiving name field value in $name variable
3. echo "Welcome, $name";
4. ?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html

```
1. <form action="login.php" method="post">
2. <table>
3. <tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
4. <tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
5. <tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
6. </table>
7. </form>
```

File: login.php

```
1. <?php
2. $name=$_POST["name"];//receiving name field value in $name variable
3. $password=$_POST["password"];//receiving password field value in $password variable
4.
5. echo "Welcome: $name, your password is: $password";
6. ?>
```

2. What is Concatenating String in PHP?

Answer:

Concatenation is an unwieldy term but a useful concept. It refers to the appending of one item onto another. Specifically, in programming, you concatenate strings . The period (.) is the operator for performing this action, and it's used like so:

```
$s1 = 'Hello, ';
$s2 = 'world!';
$greeting = $s1 . $s2;
```

php compensation operator is used to combine character string together. The php concatenation operator(.) using combine to string value to create one string.

```
<?php
   $name="John";
   $lastName="Travolta";
   echo $name." ".$lastName; // Outputs John Travolta

   $a="Hello";
   $a .= " John!";
   echo $a; // Outputs Hello John!
?>
```

3. What is New Line Feed in PHP?

Answer:

A common question beginning PHP developers have involves handling newlines in strings. The text area form element allows a user to enter text over multiple lines by pressing Return/Enter. Each use of Return/Enter equates to a newline in the resulting string. These newlines work within a text area but have no effect on rendered PHP.

nl2br — Inserts HTML line breaks before all newlines in a string

Returns **`string`** with `<br />` or `<br>` inserted before all newlines (\r\n, \n\r, \n and \r).

```
<?php
echo nl2br("foo isn't\n bar");
?>
```

The above example will output:

```
foo isn't<br />
 bar
```

4. What is decision making in PHP?

Answer:

php conditional statment allow you to make a decision, based upon the result of condition .These statment are called decision making statment.

There are following types of decision making statements available in PHP:

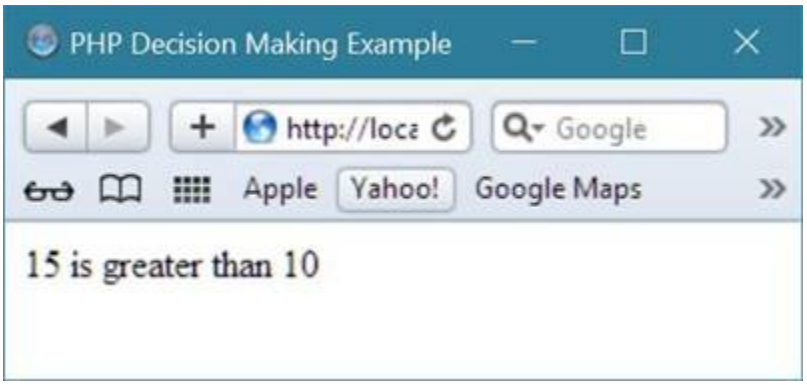
- if Statement
- if-else Statement
- if-elseif-else Statement
- switch Statement

PHP Decision Making Example

Here is an example demonstrating how decision making statements are used in PHP program to control the program.

```
<html>
<head>
	<title>PHP Decision Making Example</title>
</head>
<body>
<?php
	# this is an example on decision making statements
	# in PHP - CodesCracker
	$num1 = 5;
	$num2 = 10;
	$num3 = 15;
	if($num1>$num2)
	{
		echo "5 is greater than 10";
	}
	if($num3>$num2)
	{
		echo "15 is greater than 10";
	}
?>
</body>
</html>
```

Here is the sample output produced by the above decision making example in PHP:



5. What is crypt() in PHP?

Answer:

The crypt() is used to encrypts a string using DES, Blowfish, and MD5 (if available) algorithms.

This function behaves different on different operating systems. PHP checks what algorithms are available and what algorithms to use when it is installed.

The salt parameter is optional. However, crypt() creates a weak password without the salt. Make sure to specify a strong enough salt for better security.

There are some constants that are used together with the crypt() function. The value of these constants are set by PHP when it is installed.

Version:

(PHP 4 and above)

Syntax:

```
crypt(string1, salt)
```

Parameters:

Name	Description	Required / Optional	Type
string1	The string to be encrypted.	Required	String
salt	An optional salt string to base the hashing on. If not provided, the	Optional	String

	behavior is defined by the algorithm implementation and can lead to unexpected results.		

Return value:

The encrypted string.

Value Type: string

Example:

```
<?php

echo "Standard DES: ".crypt("Thank you")."\n<br />";

echo "Extended DES: ".crypt("Thank you")."\n<br />";

echo "MD5: ".crypt("Thank you")."\n<br />";

echo "Blowfish: ".crypt("Thank you");

?>
```

Output:

```
Standard DES: $1$cx1./y3.$H.8Trcy6pLgimqOWmGYrh/
Extended DES: $1$aU0.bl3.$h0A8HqJMF8gA3KwoZa6vq0
MD5: $1$Ic4.x85.$VmsInH4NRIb9WS5ofMGi80
Blowfish: $1$m00.1U3.$8BdJ6KtYIhRSMSJVqQpN71
View the example in the browser
```

6. What is Strcmp() in PHP?

Answer:

The strcmp() function compares two strings.

The strcmp() function is binary-safe and case-sensitive.

This function is similar to the strncmp() function, with the difference that you can specify the number of characters from each string to be used in the comparison with strncmp().

Example

Compare two strings (case-sensitive):

```
<?php
echo strcmp("Hello world!","Hello world!");
?>
```

Technical Details

Return Value:	This function returns: • 0 - if the two strings are equal • <0 - if string1 is less than string2 • >0 - if string1 is greater than string2
PHP Version:	4+

More Examples

Example 1

Compare two strings (case-sensitive = Hello and hELLo will not output the same):

```
<?php
echo strcmp("Hello","Hello");
echo "<br>";
echo strcmp("Hello","hELLo");
?>
```

7. What is strstr() in PHP?

Answer:

The strstr() function searches for the first occurrence of a string inside another string. This function is binary-safe. This function is case-sensitive. For a case-insensitive search, use stristr() function.

Syntax

strstr(*string,search,before_search*)

Parameter	Description
string	Required. Specifies the string to search
search	Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number
before_search	Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the *search* parameter.

Example

Find the first occurrence of "world" inside "Hello world!" and return the rest of the string:

```
<?php
echo strstr("Hello world!","world");
?>
```

Output:

world!

8. What is strops() in PHP?

Answer:

The strpos() function finds the position of the first occurrence of a string inside another string. The strpos() function is case-sensitive. This function is binary-safe.

Related functions:

- strrpos() - Finds the position of the last occurrence of a string inside another string (case-sensitive)
- stripos() - Finds the position of the first occurrence of a string inside another string (case-insensitive)
- strripos() - Finds the position of the last occurrence of a string inside another string (case-insensitive)

Syntax

strpos(*string,find,start*)

Parameter	Description
string	Required. Specifies the string to search

find	Required. Specifies the string to find
start	Optional. Specifies where to begin the search

Technical Details

Return Value:	Returns the position of the first occurrence of a string inside another string, or FALSE if the string is not found. Note: String positions start at 0, and not 1.
PHP Version:	4+

Example

Find the position of the first occurrence of "php" inside the string:

```
<?php
echo strpos("I love php, I love php too!","php");
?>
```

output:

7

9. What is syntax of if.. else in PHP?

Answer:

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;
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```
<?php
$t = date("H");

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

```
}
?>
```

Output:

Have a good day!

10. What is asort() in PHP?

Answer:

The asort() function is used to sorts an array. The function maintains index association.

This function is used mainly when sorting associative arrays where the actual element order is significant. This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant.

Version:

(PHP 4 and above)

Syntax:

```
asort(array_name, sort_type)
```

Parameters:

Name	Description	Required / Optional	Type
array_name	Specifies the name of the array to sort.	Required	Array
sort_type	Sets the sorting behavior. Possible type : SORT_REGULAR - Compare items normally. SORT_NUMERIC - Compare items numerically. SORT_STRING - Compare items as strings. SORT_LOCALE_STRING - compare items as strings, based on the current locale	Optional	Integer

Return value:

TRUE on success or FALSE on failure.

Value Type: Boolean.

Example:

```
<?php
$subject = array('d' => 'Language',  'c' => 'Math',  'a' => 'Science', 'b'=> 'Geography');
asort($subject);
foreach ($subject as $key => $val)
{
echo "$key = $val <br />";
}
?>
```

Output:

b = Geography
d = Language
c = Math
a = Science

11. What is ksort() in PHP?

Answer:

The ksort() function sorts an associative array in ascending order, according to the key. Use the krsort() function to sort an associative array in descending order, according to the key. Use the asort() function to sort an associative array in ascending order, according to the value.

Syntax

ksort(*array,sortingtype*);

Parameter	Description
Array	Required. Specifies the array to sort
sortingtype	Optional. Specifies how to compare the array elements/items. Possible values: • 0 = SORT_REGULAR - Default. Compare items normally (don't change types) • 1 = SORT_NUMERIC - Compare items numerically • 2 = SORT_STRING - Compare items as strings • 3 = SORT_LOCALE_STRING - Compare items as strings, based on current locale • 4 = SORT_NATURAL - Compare items as strings using natural ordering • 5 = SORT_FLAG_CASE -

Technical Details

Return Value:	TRUE on success. FALSE on failure
PHP Version:	4+

Example

Sort an associative array in ascending order, according to the key:

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

<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
ksort($age);

foreach($age as $x=>$x_value)
  {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
  }
?>

</body>
</html>
```

output:

Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35

12. What is index array in PHP?

answer:

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

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

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

In PHP, the array() function is used to create an array:

array();

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

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

or the index can be assigned manually:

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

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

output:

I like Volvo, BMW and Toyota.

UNIT – III

Q 1) What is a CSS template?
Ans:- A CSS Template is a website design created by using Cascading Style Sheets (CSS) technology. Cascading styles sheets allow web developers to easily format and style all the pages of a website at one time. CSS will be used even more because it is seen in the same way by all (99.98%) browsers.

Q 2)What is include in PHP?
Ans:- The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

Q 3) What is require in PHP?
Ans:-*equire* is identical to include except upon failure it will also produce a fatal `E_COMPILE_ERROR` level error. In other words, it will halt the script whereas include only emits a warning (`E_WARNING`) which allows the script to continue.

Q 4)What is date function?
Ans:- The date functions allow you to get the date and time from the server where your PHP script runs. You can then use the date functions to format the date and time in several ways.These functions depend on the locale settings of your server. Remember to take daylight saving time and leap years into consideration when working with these functions.

Q 5)What is constants?
Ans:- A constant is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script (except for magic constants, which aren't actually constants). A constant is case-sensitive by default. By convention, constant identifiers are always uppercase.

Q 6)What is Y,y and n in date() function?
Ans:-
The date() function formats a local date and time, and returns the formatted date string. Returns a string formatted according to the given format string using the given integer `timestamp` or the current time if no timestamp is given. In other words, `timestamp` is optional and defaults to the value of time().

Y - A four digit representation of a year.
y - A two digit representation of a year.
n - A numeric representation of a month, without leading zeros (1 to 12).

Q 7)What is cookies in PHP?

Ans:- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Q 8)What is session in PHP?

Ans:- 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, favoritecolor, etc). By default, session variables last until the user closes the browser.

Q 9)What is function call in PHP?

Ans:-PHP user define Function. Besides the built-in PHP functions, we can create our own function.Afunction is a block of statement that can be used repeatedly in a program.A function will not execute immediately when a page loads. A function will be executed by a call to the function.

Q 10)What is return statement in function?

Ans:-The return statement terminates the execution of a function and returns control to the calling function. Execution resumes in the calling function at the point immediately following the call.

A return statement can also return a value to the calling function .

Unit 4

2 Marks

1.**What is file permission?**

Ans:-You will learn how to deal with php file permissions including checking and changing file permission .file permissions specify what you can do with a particular file is system the eg:-reading , , writing executing the file.

2.What is fopen () function?

Ans:- The fopen() function opens a file or URL. If fopen() fails, it returns FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

3.**What is fread () function?**

Ans:-The **fread**() reads from an open file. The **function** will stop at the end of the file or when it reaches the specified length, whichever comes first. This **function** returns the read string, or FALSE on failure.

4.What is fclose() fuction?

Ans:-Definition and Usage. The **fclose**() **function** closes an open file. This **function**returns TRUE on success or FALSE on failure.

5.What is MySQL?

Ans:-**PHP** is the most popular scripting language for web development. It is free, open source and server-side (the code is executed on the server). **MySQL** is a Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

6.What is connect in MySQL?
Ans:- PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

7.What is use command in MySQL ?

Ans:-Creating a database in **MySQL** doesn't select it for **use**. You have to indicate it with the **USE command**. The **USE command** is also **used** when you have more than one database on a **MySQL** server and need to switch between them.

8. Write down the syntax of create database command.

Ans:-The CREATE DATABASE statement is used to create a new SQL database. A database consists of one or more tables.

You will need special CREATE privileges to create or to delete a MySQL database.

CREATE DATABASE *databasename*;

9**.What is null MySQL.**

Ans:-3.3.4.6 Working with **NULL** Values. The **NULL** value can be surprising until you get used to it. Conceptually, **NULL** means “a missing unknown value” and it is treated somewhat differently from other values. ... In **MySQL**, 0 or **NULL** means false and anything else means true.

10:-What is auto_INCREAMENT in MqSQL.

Ans:-MySQL uses the AUTO_INCREMENTkeyword to perform and auto_incremenntfeature..Bydeafault the stating value for AUTO_INCREMENT is 1 and it willincrement by one for each new record the SQL statment insert a new record into the “persone” table .The”ID” column would be assigned in a unique value.

Part - B (Each question carries Three marks) UNIT – I

Q 1) Differentiate between Print and Echo.
Ans:- echo

1. echo is a statement i.e used to display the output. it can be used with parentheses echo or without parentheses echo.
2. echo can pass multiple string separated as (,)
3. echo doesn't return any value
4. echo is faster then print

Print

1. Print is also a statement i.e used to display the output. it can be used with parentheses print( ) or without parentheses print.
2. using print can doesn't pass multiple argument
3. print always return 1
4. it is slower than echo

Q 2) How we give the comments in PHP?
Ans:- A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code.
Comments can be used to:

- Let others understand what you are doing
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

Q 3) Write and explain PHP program syntax.
Ans:- A PHP script is executed on the server, and the plain HTML result is sent back to the browser.A PHP script can be placed anywhere in the document.A PHP script starts with <?php and ends with ?>:
Syntax:-

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

The default file extension for PHP files is ".php".A PHP file normally contains HTML tags, and some PHP scripting code.

Q 4) How to create variable in PHP?
Ans:-Variable is a symbol or name that stands for a value. Variables are used for storing values such as numeric values, characters, character strings, or memory addresses so that they can be used in any part of the program.All variables in PHP start with a $ (dollar) sign followed by the name of the variable.A valid variable name starts with a letter (A-Z, a-z) or underscore (_), followed by any number of letters, numbers, or underscores.If a variable name is more than one word, it can be separated with an underscore (for example $employee_code instead of $employeecode).'$' is a special variable that can not be assigned.

Q 5) What is GET method?

Ans:- Note that the query string (name/value pairs) is sent in the URL of a GET request:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

Q 6) What is POST method?
Ans:-Note that the query string (name/value pairs) is sent in the HTTP message body of a POST request:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

Q 7) What is debugging in PHP?
Ans:- Programming code might contain syntax errors, or logical errors.Many of these errors are difficult to diagnose.Often, when programming code contains errors, nothing will happen. There are no error messages, and you will get no indications where to search for errors.Searching for (and fixing) errors in programming code is called code debugging.

Q 8) How to send html to browser?
Ans:-

Every now and again we web designers need to pass user input between pages. It could be a username or affiliate code that needs adding into a set of links; it could be details of a contact form that need passing over to a confirmation page before being sent to a database or email server; or it could be any other bit of information that a site visitor enters into one form that must be reproduced or used somewhere within another page or set of pages. Luckily, for those of us who haven't studied PHP, the recipe for passing information between pages contains few ingredients and is easy to follow and doesn't require any real knowledge of PHP.

This short tutorial will take you through the process of collecting information from visitors and sending that information to another webpage for further usage or display. We will use two recipes:

1. The first recipe uses only one PHP page and will help you see how the process works. It will not be explained in detail.
2. The second recipe is split over two pages and will be explained in detail.

The Ingredients

- 1 HTML Page
- 1 PHP Page
- 3 Dashes of HTML

-
- <form>
- <input>
- 2 Dashes of PHP
- ECHO
- $_REQUEST
- Internet Server Space or a PHP Environment (you could use LAMP)

One note of caution: the code I'm about to show is not built with security in mind so if security is a concern then you will need to encrypt and sanitize the user input.

First the recipes and then an explanation of the ingredients.

Recipe One: Putting Everything On One Page

1. Create a php page called test.php
2. Put the following lines of code into it

   PHP

```
<?php
$NAME="ENTER NAME HERE";
$AGE="ENTER AGE HERE";
$HOME="ENTER HOME TOWN HERE";
echo <<<TEXT
<p>Hello, my name is $NAME, I am $AGE years old and live in $HOME and I am learning how to use PHP to create typing shortcuts by passing information from one place to another place within my web pages.</p>
TEXT;
?>
```

3. Fill out the details for $NAME, $AGE and $HOME with your name, age and hometown then save the page before opening it in a web browser.

Quick Explanation:

When you view the page you will see that $NAME, $AGE and $HOME between the and elements are replaced with the details you placed between the quotation marks. For example in $NAME="ENTER NAME HERE", anything that replaces "ENTER NAME HERE" will replace $NAME.

$NAME, $AGE and $HOME are called variables and can be assigned any single character or character string value. The echo command prints data to the screen. Our echo command is special because it has a start and end point as determined by >>>TEXT and TEXT, respectively. Whatever is written between those start (>>>TEXT) and end (TEXT) points is echoed to the

screen. Variables like our $NAME, $AGE and $HOME are replaced with their assigned values when echo prints them to the screen.

Recipe Two: Passing information from one page to another page

- Create an html page called page-one.html,
- Put the following lines of code into it

PHP

```
1  <html>
2  <head></head>
3  <body>
4  <form id="Form" action="page-two.php" method="post">
5  <p>Enter Name: <input type="text" name="Name" /></p>
6  <p>Enter Age: <input type="text" name="Age" /></p>
7  <p>Enter Hometown: <input type="text" name="Town" /></p>
8  <p><input type="Submit" name="Form_Submit" value="Send"
   /></p>
9  </form>
10 </body>
11 </html>
```

- Create a PHP page called page-two.php
- Put the following lines of code into it

PHP

```
1 <?php
2 $NAME=$_REQUEST['Name'];
3 $AGE=$_REQUEST['Age'];
4 $HOME=$_REQUEST['Town'];
5 echo <<<TEXT
6 <p>Hello, my name is <b>$NAME</b>, I am <b>$AGE</b> years old and live in
7 <b>$HOME</b> and I am learning how to use PHP to create typing shortcuts by
  passing information from one place to another place within my web pages.</p>
8 TEXT;
```

?>

- View page-one.html in your web browser, fill in the requested details then press send.

Explanation

As happened in Recipe One, $NAME, $AGE and $HOME are replaced by the data they represent but this time their assigned values come from the details entered into the form on page-one.html – page-one.html collects information and passes it to page-two.php . This is done in four steps:

1. The <form> element has an Action and Method parameter e.g <form id="FormOne" action="page-two.php" method="post">
    - Action points to the page where the data collected by the form will be sent to be processed
    - Method tells the browser how to send the collected data. It takes one of two values: get or post
        1. get sends data via the URL
        2. post sends data via HTTP
    - Anything sent by get is visible in a browser's address bar (i.e within the URL) whereas anything sent via post is only visible to hackers. Get can only handle limited character strings i.e only characters that can be embedded within a URL and only as many characters as a URL can carry. Conversely, post has no such character limitations. More often than not, post is the best method to use when sending data between pages.
2. Every <input> element has a name attribute which identifies the data entered into its form field so it can be requested by page-two.php e.g <input type="text" name="Name" />
3. When the "Send" button on page-one.html is activated the data from the form is posted to page-two.php where it is requested by name using the $_REQUEST['name'] command and is assigned to a variable with an equality (=) symbol. For example the instruction $NAME=$_REQUEST['Name'] assigns the variable $NAME with the value entered into <input type="text" name="Name" />
4. The data is then printed to the screen using the echo command to display whatever is placed between >>>TEXT and TEXT and the variables between those delimiters are swapped for whatever values they have been assigned.

The $_REQUEST command can be replaced with $_GET or $_POST. You would use $_GET when the form method is get and $_POST when the form method is post. $_REQUEST catches information sent either way and is useful for lazy people like me.

Working example of Recipe Two

How To Use It

For those who have no understanding of PHP and who do not want to know how it works

First, in the HTML page, create an input field by using the HTML <input /> element and give it a unique name. Put the <input /> element between the <form></form> tags. For example, Name a fruit: <input type="text" name="fruit" />

Next, create a variable (like $NAME) between <?php and echo in the PHP page and assign it a value by relating it to a corresponding unique name from one of your <input /> elements. For example, $FRUIT=$_REQUEST[‘fruit’];

Lastly, call it by placing it in whatever text or HTML is put between the echo <<<TEXT and TEXT statements (a variable is like a pronoun, it stands in for something else. All variables begin with a dollar ($) sign). For example, My favorite fruit is $FRUIT

Be aware that PHP differentiates between lowercase and uppercase letters so fruit is not the same as FRUIT hence I’ve been able to use the word “fruit” twice in the above examples to mean two different things.

Safety First (or second, or last)

A few notes of caution on PHP usage:

You should never trust the data entered by a site’s visitor – treat it as you would a plague, check that the data entered fits the format required by the form field. For example, a telephone number will always contain numbers, could contain a dot or plus sign so strip it of any other characters;

When you use a double-quote (“) within an echo statement, you should place a backslash ( ) in front of it otherwise you will get an error message instead of your intended text message. For example

echo “this is a “double quote” within an echo statement”, should be written

echo “this is a “double quote” within an echo statement”

Q 9) Write Characteristics of PHP?

Ans:- Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Q 10) What is PHP Parser?

Ans:- To parse, in computer science, is where a string of commands – usually a program – is separated into more easily processed components, which are analysed for correct syntax and then attached to tags that define each component. The computer can then process each program chunk and transform it into machine language.

Part - B (Each question carries Three marks) UNIT – II

Unit 2

3marks

1.Wrte a php program to con concatenate two string.

Ans:-<php

```
$str1="hello php";
$str2="hello php works?";
echo $str1." ".$str2;
?>
```

2.Explain any three string function in php.

Ans:-1.php strtolower ():-The strlower () function return string to lowercase letter.
2.phpstrtoupper ():-The string strtoupper () function return string in uppercase letters.
3.phpUcfirst ():-The Ucfirst () function return string converting first character into appear. It does change the can of other change.

3.Explainstrtolower () and strtoupper () in php.

Ans:-1.php strtolower ():-The strlower () function return string to lowercase letter.
2.phpstrtoupper ():-The string strtoupper () function return string in uppercase letters.

4.Write a note on if...else statment?

Ans:-php conditional statment .Very often when you write code, you want to perform different action for diffrentconditions.you can use conditional statments your code to do this if ....else statment execute some code if a condition is true and another code if condition is false.

5.What is super global variable?

Ans:-several predefined variables in "superglobals" which mean that they are always accessible, regardless of scope and you can access them from any function class aur file without having to do anything special.

6.**what is nested if ....else statment ?**

Ans:- nested if statmentsmeanse if block inside another if blocks. shortly a control structure inside another control structure. hear we can see another if..else structure another the if block and else block.

7.What is function in php?

Ans:-php function is a piace of code that can be reuse many types it can take input else argument list and return value.There are thousand of built in function in php.

8.What is library function in php?

Ans:-

The Standard PHP Library (SPL) is a collection of interfaces and classes that are meant to solve common problems.

No external libraries are needed to build this extension and it is available and compiled by default in PHP 5.0.0.

SPL provides a set of standard datastructure, a set of iterators to traverse over objects, a set of interfaces, a set of standard Exceptions, a number of classes to work with files and it provides a set of functions like spl_autoload_register().

PHP comes standard with many functions and constructs. There are also functions that require specific PHP extensions compiled in, otherwise fatal "undefined function" errors will appear. For example, to use image functions such as imagecreatetruecolor(), PHP must be compiled with GD support. Or, to use mysql_connect(), PHP must be compiled with MySQL support. There are many core functions that are included in every version of PHP, such as the string and variable functions. A call to phpinfo() or get_loaded_extensions() will show which extensions are loaded into PHP. Also note that many extensions are enabled by default and that the PHP manual is split up by extension. See the configuration, installation, and individual extension chapters, for information on how to set up PHP.

Reading and understanding a function's prototype is explained within the manual section titled how to read a function definition. It's important to realize what a function returns or if a function works directly on a passed in value. For example, str_replace() will return the modified string while usort() works on the actual passed in variable itself. Each manual page also has specific information for each function like information on function parameters, behavior changes, return values for both success and failure, and availability information. Knowing these important (yet often subtle) differences is crucial for writing correct PHP code.

9.What is empty() in php?

Ans:-php has different functions which can be used to test the value of variable. Three useful functions for this are isset(), emty() and i_null().all these function return a booleanvalue.If these function are not used in correct way they can cause unexpected result.

10.What is associative array in php?

Ans:-We can associate name with each array elements in php using (==>).In the product array,We allowed php to give each item the default index. This meant the first item,we added became 0,The second item 1, and so on.php also supports associative array,In an associative array,we can associate any key or index we want with each value.

Part - B (Each question carries Three marks)

UNIT – III

1. Write a note on Template in PHP.

Answer:

So what is a template, in the first place? For our purpose, a template is an HTML-like document that defines the *presentation* of a web page, while a PHP script supplies the *content*. The separation of content and presentation is at the heart of any GUI paradigm. To see why this is desirable, consider the following hypothetical, yet realistic script. It's a script to retrieve a list of books from the database and display it as a table with alternate colours for each row, or "No books found" if the list is empty.

Basic Templating

The simplest use of templates in PHP is very powerful for reducing errors and time spent on your pages. First, make sure your server has PHP enabled, etc., etc.

When you're ready to start, make one page that will be the template for all your pages. For example:

```
This is a title at the top of my page

This is the body of my page

This is a copyright notice
```

Now, say you want 2 more pages with the same header and footer. You don't have to code it again. You can save the header as one template and the footer as another. Just take all the header html up to the part where your body text begins:

```
<html><body><p>This is a title at the top of my page</p>
```

Now save this as a separate file. I like to use the extension .inc (do the same with the footer)

```
<p>This is the bottom of my page</p></body></html>
```

Now, in your main page, just type:

```
<?php require('header.inc'); ?>
<p>this is the body of my page</p>
<?php require('footer.inc'); ?>
```

And that's your page. Save it as a .php, upload it, and check it.

Notes

You can also use the **include()** or **include_once()** functions, if the page should continue loading, even if the file(s) can not be included.

The **require()**, **include()** and **include_once()** functions will work with other file types, and can be used anywhere on a page.

Advanced: Try using this with an if statement for DYNAMIC templating... ooh...

Managed Templating

Managed Templating allows you to create and use PHP Templates with a Template Engine. The PHP Developer/Designer doesn't have to create the engine for it to be used. The most reliable PHP Templating Engine is Smarty ([[1]]). Managed Template Systems are easy to use and are mostly used in big websites because of the need for dynamic paging. MediaWiki is one example of a Managed Template System. Managed Templating is easy to use for new and advanced users, for example:

- index.php

```
// This script is based on Smarty
require_once("libs/Smarty.inc.php");
// Compiled File Directory
$smarty->compile_dir = "compiled";
// Template Directory
$smarty->template_dir = "templates";
// Assign a Variable
$smarty->assign("variable","value");
// Display The Parsed Template
$smarty->display("template.tpl");
```

- template.tpl

```
The Value of Variable is : {$variable}
```

- Output

```
The Value of Variable is : value
```

Roll Your Own

Template engines work great, however if you are just looking for the basic Search and Replace template functionality, writing your own script is a snap.

- Simple template function

```
function Template($file, $array) {
	if (file_exists($file)) {
		$output = file_get_contents($file);
		foreach ($array as $key => $val) {
			$replace = '{'.$key.'}';
			$output = str_replace($replace, $val, $output);
		}
		return $output;
	}
}
```

- Using the function

```
$fruit = 'Watermelon';
$color = 'Gray';

//parse and return template
$Template_Tpl = Template('template.tpl',
	array
	(
		'fruit'	=> $fruit,
		'color'	=> $color
	));

//display template
```

```
echo $Template_Tpl;
```

- template.tpl, Template used for the above function

```
<p>
  <b>Your Favorite Food Is: {fruit}</b>
  <b>Your Favorite Color Is: {color}</b>
</p>
```

- Parsed template output

```
<p>
  <b>Your Favorite Food Is: Watermelon</b>
  <b>Your Favorite Color Is: Gray</b>
</p>
```

2. Write a note on mail() function.

Answer:

Definition and Usage

The mail() function allows you to send emails directly from a script.

Syntax

mail(*to,subject,message,headers,parameters*);

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. **Windows note:** If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php $txt = str_replace("\n.", "\n..", $txt); ?>
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n).

	Note: When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

Technical Details

Return Value:	Returns the hash value of the *address* parameter, or FALSE on failure. **Note:** Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The *parameter* parameter is disabled in safe mode PHP 4.0.5: The *parameter* parameter was added

Example

Send a simple email:

```
<?php
// the message
$msg = "First line of text\nSecond line of text";

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```

Example 2

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

Example 3

Send an HTML email:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
</body>
</html>
";

// Always set content-type when sending HTML email
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

// More headers
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";

mail($to,$subject,$message,$headers);
?>
```

3. How to set Cookies in PHP?

Answer:

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
   echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
   echo "Cookie '" . $cookie_name . "' is set!<br>";
   echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

output:

Cookie named 'user' is not set! **Note:** You might have to reload the page to see the value of the cookie.

4. Write a note on delete session in PHP.

Answer:

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.

What is a PHP Session?

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.

So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a database.

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
```

```
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```
<?php
print_r($_SESSION);
?>

</body>
</html>
```

How does it work? How does it know it's me?

Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
```

```
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

5. Write a note on modify cookies in PHP.

Answer:

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Output:

Cookie 'user' is set!
Value is: John Doe

Note: You might have to reload the page to see the new value of the cookie.

6. What is session variable in PHP.

Answer:

What is a PHP Session?

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.

So; Session variables hold information about one single user, and are available to all pages in one application.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Output:

Favorite color is .
Favorite animal is .

7. Write a note delete cookies.

Answer:

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Output:

Cookie 'user' is deleted.

8. What is isset() function?

Answer:

isset() function

The isset () function is used to check whether a variable is set or not. If a variable is already unset with unset() function, it will no longer be set. The isset() function return false if testing variable contains a NULL value.

Version:

(PHP 4 and above)

Syntax:

isset(variable1, variable2.....)

Parameter:

Name	Description	Required / Optional	Type
variable1	The variable being checked	Required	Mixed*
Variable2	More variable to be checked.	Optional	Mixed*

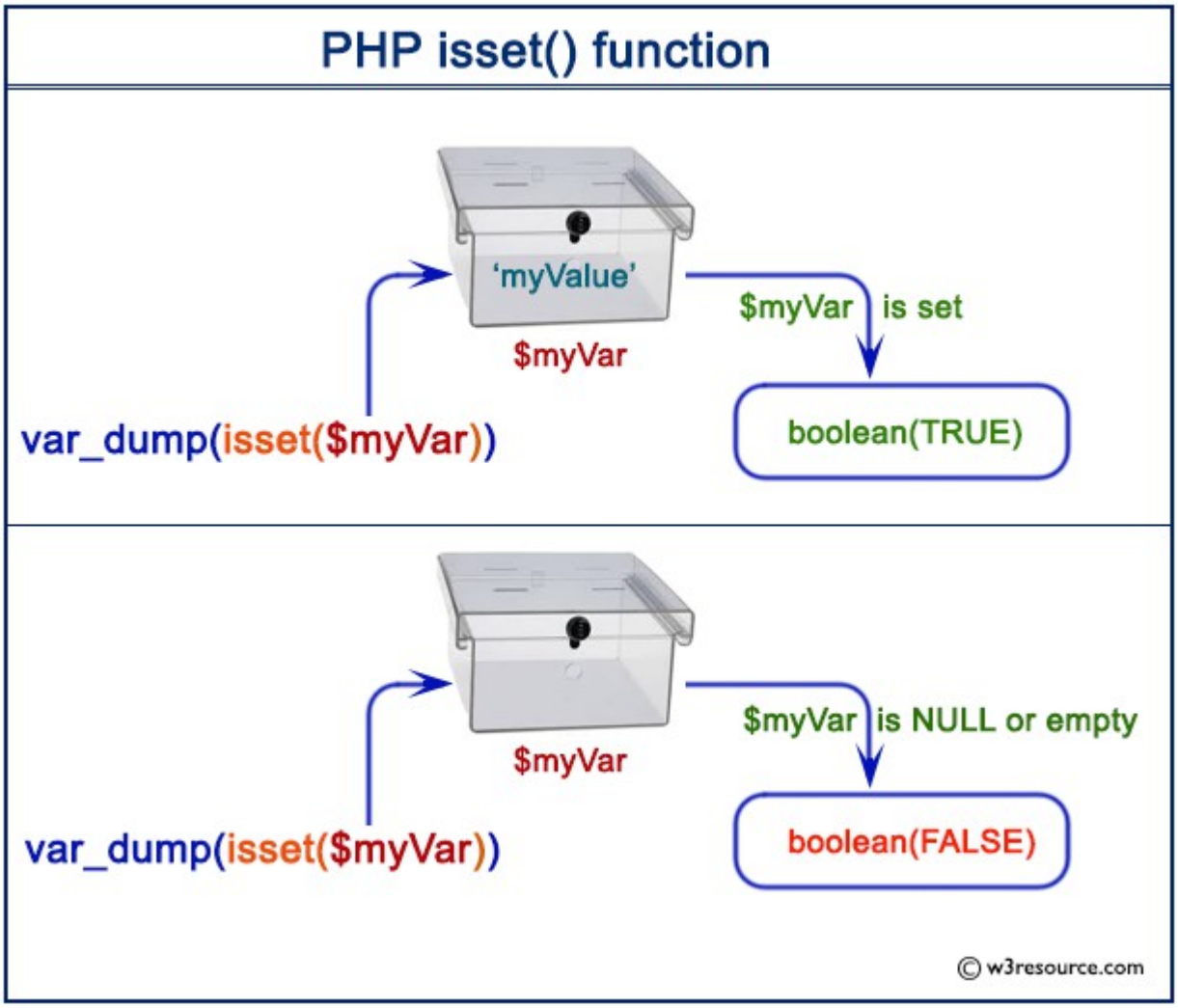
*Mixed: Mixed indicates that a parameter may accept multiple (but not necessarily all) types.

Return value:

TRUE if variable (variable1,variable2..) exists and has value not equal to NULL, FALSE otherwise.

Value Type: Boolean.

Pictorial presentation of PHP isset() function



Example:

```
<?php
$var1 = 'test';
var_dump(isset($var1));
?>
```

Copy

Output :

```
bool(true)
```

9. What is session_start() function?

Answer:

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Output:

Session variables are set.

10. What is htmlspecialchars() function?

Answer:

Definition and Usage

The htmlspecialchars() function converts some predefined characters to HTML entities.

The predefined characters are:

- & (ampersand) becomes &
- " (double quote) becomes "
- ' (single quote) becomes '
- < (less than) becomes <
- > (greater than) becomes >

Tip: To convert special HTML entities back to characters, use the htmlspecialchars_decode() function.

Syntax

htmlspecialchars(*string,flags,character-set,double_encode*)

Parameter	Description
string	Required. Specifies the string to convert
flags	Optional. Specifies how to handle quotes, invalid encoding and the used document type. The available quote styles are: • ENT_COMPAT - Default. Encodes only double quotes • ENT_QUOTES - Encodes double and single quotes • ENT_NOQUOTES - Does not encode any quotes Invalid encoding: • ENT_IGNORE - Ignores invalid encoding instead of having the function return an empty string. Should be avoided, as it may have security implications. • ENT_SUBSTITUTE - Replaces invalid encoding for a specified character set with a Unicode Replacement Character U+FFFD (UTF-8) or &#FFFD; instead of returning an empty string. • ENT_DISALLOWED - Replaces code points that are invalid in the specified doctype with a Unicode Replacement Character U+FFFD (UTF-8) or &#FFFD; Additional flags for specifying the used doctype: • ENT_HTML401 - Default. Handle code as HTML 4.01 • ENT_HTML5 - Handle code as HTML 5 • ENT_XML1 - Handle code as XML 1 • ENT_XHTML - Handle code as XHTML
character-set	Optional. A string that specifies which character-set to use. Allowed values are:

- UTF-8 - Default. ASCII compatible multi-byte 8-bit Unicode
- ISO-8859-1 - Western European
- ISO-8859-15 - Western European (adds the Euro sign + French and Finnish letters missing in ISO-8859-1)
- cp866 - DOS-specific Cyrillic charset
- cp1251 - Windows-specific Cyrillic charset
- cp1252 - Windows specific charset for Western European
- KOI8-R - Russian
- BIG5 - Traditional Chinese, mainly used in Taiwan
- GB2312 - Simplified Chinese, national standard character set
- BIG5-HKSCS - Big5 with Hong Kong extensions
- Shift_JIS - Japanese
- EUC-JP - Japanese
- MacRoman - Character-set that was used by Mac OS

Note: Unrecognized character-sets will be ignored and replaced by ISO-8859-1 in versions prior to PHP 5.4. As of PHP 5.4, it will be ignored an replaced by UTF-8.

double_encode

Optional. A boolean value that specifies whether to encode existing html entities or not.

- TRUE - Default. Will convert everything
- FALSE - Will not encode existing html entities

Technical Details

Return Value:

Returns the converted string

If the *string* contains invalid encoding, it will return an empty string, unless either the ENT_IGNORE or ENT_SUBSTITUTE flags are set

PHP Version:

4+

Changelog:

The default value for the *character-set* parameter was changed to UTF-8 in PHP 5

ENT_SUBSTITUTE, ENT_DISALLOWED, ENT_HTML401, ENT_HTML5, ENT_XML1 and ENT_XHTML were added in PHP 5.4

ENT_IGNORE was added in PHP 5.3

The *double_encode* parameter was added in PHP 5.2.3

The *character-set* parameter was added in PHP 4.1

Example

Convert the predefined characters "<" (less than) and ">" (greater than) to HTML entities:

```
<?php
$str = "This is some <b>bold</b> text.";
echo htmlspecialchars($str);
?>
```

The HTML output of the code above will be (View Source):

```
<!DOCTYPE html>
<html>
<body>
This is some &lt;b&gt;bold&lt;/b&gt; text.
</body>
</html>
```

The browser output of the code above will be:

This is some bold text.

output:

This is some bold text.

Converting < and > into entities are often used to prevent browsers from using it as an HTML element. This can be especially useful to prevent code from running when users have access to display input on your homepage.

Part - B (Each question carries Three marks)

UNIT – IV

1.What is GRANT command.

Ans:-GRANT command is command use to provide access or priviliges on the database objects to the users .User_name is the named of the user to whom an access right is being granted .User_name is the name of the user of the to whom an access right in being granted.

2.How to create database and table MySQL.

Ans:-1.CREATE DATABASE –create the database. To use this statement , you need the CREATE privilege database.2
2.CREATE TABLE:- create the table……
3.INSERT:-To add /insert data to table i.e. insert new row into an existing table.

3.Write down MySQL _connect() function in MySQL.

Ans:-The MySQL_connect () function opens a new connection to the MySQL sever. It return recourse if connection is established or null.
Syntax:-
Resource mysqli_connect (server , username , password)

4.Wite down MySQL_query () function in MySQL .

Ans:-The mysql_query () is used to execute query on the default database.Themysqli_query () function perform a query against the database. The function returns the query handle for SELECTET quries, TRUE/FALSE for other queries () instead. This function fetch and buffers and recordset automatically. To run and buffered query. The MYSQL connection . Before performing any operation on a MySQL database, it is required to set a connection to the MySQL database you want to work with. And this is done by mysql_connection () function.

5.Write down mysql_fetch_array () function mysql.

Ans:-The mysql_fetch_array () is used to retrieve a row of data as an array from a My SQL result handle. Refers to the resource return by the valid mysql query .The type of the result is an array. An array containing one row of data , or FALSE if there are no more rows.

6.write down mysql_num_rows() function inmysql.

Ans:-The mysql_num_row () is used to get the number of row in a MySQL result handle. Reefers to the resource return by valid mysql query. Returns number of row in result on success, or null on error.

7.Write down the syntax and example insert command in mysql.

Ans:-Syntax:

INSERT INTOtable_name
(column1, column2, column3,….)
VALUES (value1 , value 2, value3…..)
Example:-

8.Write down the syntax and example update command in mysql.

Ans:-Synatx:-

```
UPDATE table_name
SET column 1 = value , column 2 =value 2,…..
WHERE
Some _column=some _value
Example:-
<?php
$servername=”localhost”;
$username = “username”;
$password =”password”;
$dbname=”myDB”;
$con=new
Mysqli($servername,$username,$password,$dbname);
If($conn->connect_error)
{
Die(“connection failed:”.$conn->connect_error);
}
$sql=”UPDATE MyGuests
SET” lastname =’Doe’;
WHERE id=2”;
If ($conn->query($sql)===TRUE)
{
Echo “Recorded updated successfully “);
{
Else
}
Echo “error updating record :” . $conn->close;
}
$conn->close();
?>
```

9.Write down the syntax and example Date() function in mysql.

Ans:-Syntax:-DATE(expressuion)

Example:-

```
<?php
$servername=”localhost”;
$username=”username”;;
$password=”password”;
```

```
$dbname=”dbname”;
Mysqli($servername,$username,$password,$dbname);
Foreach ($db->query(SELECT DATE(2008-05-17  11:31:31”) as required_DATE ‘)as $row)
{
Echo “<tr>”;
Echo “<td>”; .$row[‘required_DATE]  .”</td>”;
Echo “<tr/>”;
}
?>
```

10.Write down the syntax and example delete command in mysql.
Ans:-Syntax:-
DELETE FROM ‘table _name’ [WHERE condition];

Part - C (Each question carries Five marks)

UNIT – I
Q 1) Explain different type of variable in PHP.
Ans:- Variables are "containers" for storing information.
PHP variable ScopeIn PHP, variables can be declared anywhere in the script.The scope of a variable is the part of the script where the variable can be referenced/used.PHP has three different variable scopes:

- local
- global
- static

1)Global Variable:-A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:The global keyword is used to access a global variable from within a function.To do this, use the global keyword before the variables (inside the function):

2)Local Variable :- A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

3)Static Variable:-normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.To do this, use the static keyword when you first declare the variable:

Q 2) Explain form control in detail.
Ans:-

Form Controls

XForms User Interface controls—form controls—are declared using markup elements, and their behavior refined via markup attributes. This markup may be decorated with`class` attributes that can be used in CSS stylesheets to

deliver a customized look and feel. XForms user interface controls are bound to the underlying instance data using binding attributes as defined in the chapter **6 Constraints**.

Form controls enable accessibility by taking a uniform approach to such features as captions, help text, tabbing and keyboard shortcuts. Internationalization issues are addressed by following the same design principles as in XHTML. All form controls are suitable for styling using Aural CSS (ACSS) style properties.

Form controls encapsulate high-level semantics without sacrificing the ability to deliver real implementations. For instance, form controls `selectOne` and `selectMany` enable the user *select one or more items from a set*. These form controls distinguish the functional aspects of the underlying control from the presentational aspects (through `class`attributes) and behavior (through XForms Action elements). This separation enables the expression of the intent underlying a particular form control—see [AUI97] for a definition of such high-level user interaction primitives.

Form controls when rendered display the underlying data values to which they are bound. While the data presented to the user through a form control must directly correspond to the bound instance data, the display representation is not required to exactly match the lexical value. For example, user agents should apply appropriate conventions to the display of dates, times, durations and numeric values including separator characters.

XForms user interface controls use common attributes and elements that are defined in (**8.12 Common Markup**). Sections in this chapter define the various form controls by specifying the following:

Description
Examples
Data Binding Restrictions
Implementation Requirements
XML Representation

8.1 input

Description: This form control enables free-form data entry.

```
<input ref="order/shipTo/street" class="streetAddress">
  <caption>Street</caption>
  <hint>Please enter the number and street name</hint>
</input>
```

In the above, the `class` attribute can be used by a stylesheet to specify the display size of the form control. Note that the constraints on how much text

can be input are obtained from the underlying XForms Model definition and not from these display properties.

A graphical browser might render the above example as follows:

Data Binding Restrictions: Binds to any simpleContent (except `xsd:base64Binary`, `xsd:hexBinary` or any datatype derived from these).

Implementation Requirements: Must allow entry of a lexical value for the bound datatype. Implementations should provide the most convenient means possible for entry of datatypes and take into account localization and internationalization issues such as representation of numbers. For example, an `input` bound to an instance data node of type `Date` might provide a calendar control to enter dates; similarly, an input control bound to data instance of type `boolean` might be rendered as a simple checkbox.

XML Representation: <`input`>

```
<input
  (single node binding attributes)
  (common attributes)
  inputMode = xsd:string
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</input>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
common attributes defined in **8.12.1 Common Attributes**
inputMode - this form control accepts an input mode hint. **D Input Modes**

Note:

Notice that not binding any user interface to a piece of instance data results in an *hidden* form control in XForms; consequently, there is no need to explicitly define input form controls with `type="hidden"` as in HTML.

8.2 secret

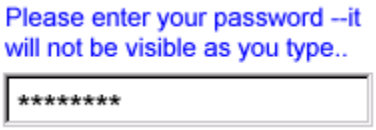
Description: This form control is used for entering information that is considered sensitive, and thus not echoed to a visual or aural display as it is being entered, e.g., password entry.

Password Entry

```
<secret ref="/login/password">
  <caption>Password</caption>
```

```
  <hint>Please enter your password --it will not
    be visible as you type.</hint>
</secret>
```

A graphical browser might render this form control as follows:



Data Binding Restrictions: Identical to `input`.

Implementation Requirements: In general, implementations, including accessibility aids, must render a "*" or similar character instead of the actual characters entered, and thus must not render the entered value of this form control. Note that this provides only a casual level of security; truly sensitive information will require additional security measures outside the scope of XForms.

XML Representation <`secret`>

```
<secret
  (single node binding attributes)
  (common attributes)
  inputMode = xsd:string
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</secret>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
common attributes defined in **8.12.1 Common Attributes**
inputMode - this form control accepts an input mode hint. **D Input Modes**

8.3 textarea

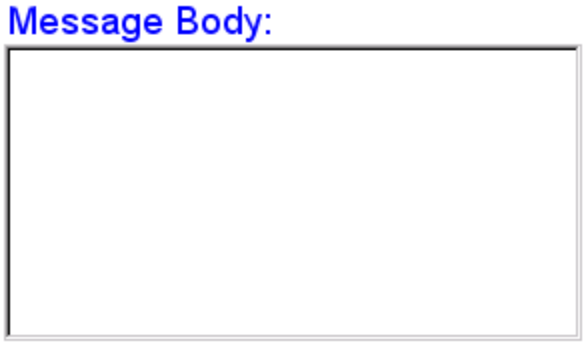
Description: This form control enables free-form data entry and is intended for use in entering multiline content, e.g., the body of an email message.

Email Message Body

```
<textarea ref="message/body" class="messageBody">
  <caption>Message Body</caption>
  <hint>Enter the text of your message here</hint>
</textarea>
```

In the above, the `class` attribute can be used by a stylesheet to specify the display size of the form control. Note that the constraints on how much text can be input are obtained from the underlying XForms Model definition and not from these display properties.

A graphical browser might render the above example as follows:



Data Binding Restrictions: Binds to `xsd:string` or any derived simpleContent.

Implementation Requirements: Must allow entry of a lexical value for the bound datatype, including multiple lines of text.

XML Representation: <`textarea`>

```
<textarea
  (single node binding attributes)
  (common attributes)
  inputMode = xsd:string
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</textarea>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
common attributes defined in **8.12.1 Common Attributes**
inputMode - this form control accepts an input mode hint. **D Input Modes**

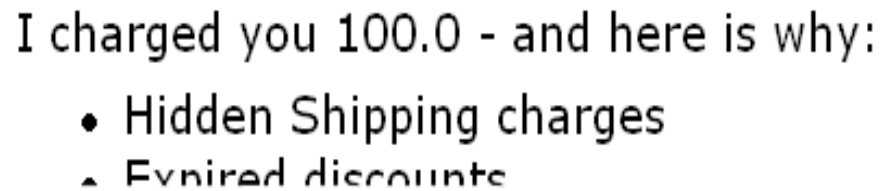
8.4 output

Description: This form control renders a value from the instance data, but provides no means for entering or changing data. It is typically used to display values from the instance, and is treated as `display:inline` for purposes of layout.

Explanatory Message

```
I charged you -
<output ref="order/totalPrice"/>
and here is why:
```

A graphical browser might render an output form control as follows:



Data Binding Restrictions: Binds to any simpleContent.

Implementation Requirements: Must allow display of a lexical value for the bound datatype. Implementations should provide the most convenient means possible for display of datatypes and take into account localization and internationalization issues such as representation of numbers.

XML Representation: <`output`>

```
<output
  (single node binding attributes)
>
  <!-- empty content -->
</output>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**

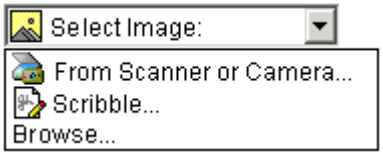
8.5 upload

Description: This form control enables the common feature found on Web sites to upload a file from the local file system, as well as accepting input from various devices including microphones, pens, and digital cameras.

Uploading An Image

```
<upload ref="mail/attach1" mediaType="image/*">
  <caption>Select image:</caption>
</upload>
```

A graphical browser might render this form control as follows:



Data Binding Restrictions: This form control can only be bound to datatypes `xsd:base64Binary` or `xsd:hexBinary`, or types derived by restriction from these.

Implementation Requirements: For suitable mediaTypes:

- Implementations with a file system should support *file upload*—selecting a specific file. The types of files presented by default must reflect the

mediaType specified in the XForms Model, for example defaulting to only audio file types in the file dialog when the mediaType is "audio/*". In XForms 1.0, there is a 1:1 binding between a upload form control and one of the `binary` datatypes, although that single file may be compound (e.g. application/zip).

- Implementations with specific pen/digitizer hardware should (and implementations with other pointing devices may) support *scribble*—allowing in-place creation of pen-based data.
- Implementations with specific audio recording capabilities should support *record audio*—in-place recording of an audio clip.
- Implementations with a digital camera/scanner interface or screen capture should support *acquire image*—in-place upload of images from an attached device.
- Implementations with video recording capability should provide a *record video* option.
- Implementations with 3d capabilities should provide a 3d interface option.
- Implementations may provide proprietary implementations (for example, a mediaType of text/rtf could invoke an edit window with a proprietary word processing application)
- Implementations are encouraged to support other input devices not mentioned here.
- Implementations which cannot support upload for the given mediaType must make this apparent to the user.

XML Representation: <`upload`>

```
<upload
  (single node binding attributes)
  (common attributes)
  mediaType = list of content types
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</upload>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
common attributes defined in **8.12.1 Common Attributes**
mediaType - list of suggested media types, used by the XForms Processor to determine which input methods apply.

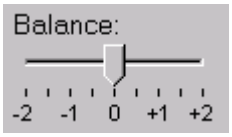
8.6 range

Description: This form control allows selection from a continuous range of values.

Picking From A Range

```
<range ref="/stats/balance" start="-2.0" end="2.0" stepSize="0.5">
  <caption>Balance</caption>
</range>
```

A graphical browser might render this as follows:



Data Binding Restrictions: Binds only the following list of datatypes, or datatypes derived by restriction from those in the list: `xsd:duration`, `xsd:date`, `xsd:time`, `xsd:dateTime`, `xsd:gYearMonth`, `xsd:gYear`, `xsd:gMonthDay`, `xsd:gDay`, `xsd:gMonth`, `xsd:float`, `xsd:decimal`, `xsd:double`.

Implementation Requirements: Must allow input of a value corresponding to the bound datatype. Implementations should inform the user of the upper and lower bounds, as well as the step size, if any. In graphical environments, this form control may be rendered as a "slider" or "rotary control".

Notice that the attributes of this element encapsulate sufficient metadata that in conjunction with the type information available from the XForms Model proves sufficient to produce meaningful prompts when using modalities such as speech, e.g., when using an accessibility aid. Thus, an aural user agent might speak a prompt of the form *Please pick a date in the range January 1, 2001 through December 31, 2001.*

XML Representation: <`range`>

```
<range
  (single node binding attributes)
  (common attributes)
  start = datavalue
  end = datavalue
  stepSize = datavalue-difference
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</range>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
common attributes defined in **8.12.1 Common Attributes**
start - optional hint for the lexical starting bound for the range—a legal value for the underlying data.

end - optional hint for the ending bound for the range—a legal value for the underlying data.
stepSize - optional hint to use for incrementing or decrementing the value. Should be of a type capable of expressing the difference between two legal values of the underlying data.

8.7 button

Description: This form control is similar to the HTML element of the same name and allows for user-triggered actions. This form control may also be used to advantage in realizing other custom form controls.

Simple Button

```
<button>
  <caption>Click here</caption>
</button>
```

Data Binding Restrictions: Binding not possible for this form control.

Implementation Requirements: The user agent must provide a means to generate an `xforms:activate` event on the form control. Graphical implementations would typically render this form control as a push-button with the caption on the button face. Stylesheets can be used to style the button as an image.

XML Representation: <`button`>

```
<button
  (common attributes)
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</button>
```

common attributes defined in **8.12.1 Common Attributes**

8.8 submit

Description: This form control initiates submission of all or part of the instance data to which it is bound.

Submit

```
<submit submitInfo="timecard">
  <caption>Submit Timecard</caption>
</submit>
```

Data Binding Restrictions: Binding not possible for this form control.

Implementation Requirements: Upon receiving event `xforms:activate`, this form control dispatches event `xforms:submit` to the `submitInfo` element specified by required attribute `submitInfo`.

XML Representation: <submit>

```
<submit
  (common attributes)
  submitInfo = xsd:IDREF #REQUIRED
>
  <!-- caption, (help|hint|alert|action|extension)* -->
</submit>
```

submitInfo - Required reference to element `submitInfo`
common attributes defined in **8.12.1 Common Attributes**

8.9 selectOne

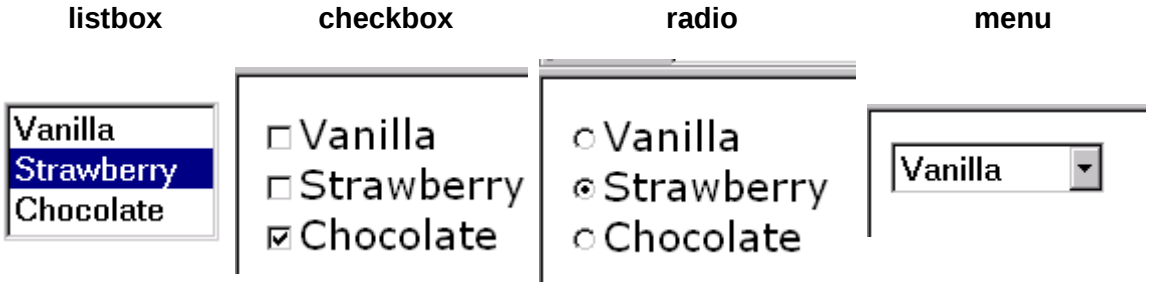
Description: This form control allows the user to make a single selection from multiple choices.

Pick A Flavor

```
<selectOne ref="my:icecream/my:flavor">
  <caption>Flavour</caption>
  <item>
    <caption>Vanilla</caption>
    <value>v</value>
  </item>
  <item>
    <caption>Strawberry</caption>
    <value>s</value>
  </item>
  <item>
    <caption>Chocolate</caption>
    <value>c</value>
  </item>
</selectOne>
```

In the above example, selecting one of the choices will result in the associated value given by element `value` on the selected item being set in the underlying data instance at the location `icecream/flavor`.

A graphical browser might render this form control as any of the following:



Data Binding Restrictions: Binds to any simpleContent.

Implementation Requirements: The caption for each choice must be presented, allowing at all times exactly one selection. This form control stores the value corresponding to the selected choice in the location addressed by attribute `ref`. The value to be stored is either directly specified as the contents of element `value`, or specified indirectly through attribute `ref` on element `value`.

Note that the datatype bound to this form control may include a non-enumerated value space, e.g., `xsd:string`. In this case, control `selectOne` may have attribute`selection="open"`. The form control should then allow free data entry, as described in **8.1 input**.

For closed selections:If the initial instance value matches the storage value of one of the given items, that item is selected. If there is no match, the first item is initially selected.

For open selections: If the initial instance value matches the storage value specified by one of the items, the first such matching item is selected. Otherwise, the selected value is the initial lexical value. Free entry text is handled the same as form control `input` **8.1 input**.

User interfaces may choose to render this form control as a pulldown list or group of radio buttons, among other options. The selectUI attribute offers a hint as to which rendering might be most appropriate, although any styling information (such as CSS) should take precedence.

Typically, a stylesheet would be used to determine the exact appearance of form controls, though a means is provided to suggest an appearance through attribute `selectUI`. The value of the attribute consists of one of the following values, each of which may have a platform-specific look and feel.

radio
checkbox
menu
listbox

XML Representation: <`selectOne`>

```
<selectOne
  (single node binding attributes)
  (common attributes)
  selectUI = ("radio" | "checkbox" | "menu" | "listbox" )
  selection = "open" | "closed" : "closed"
>
  <!-- caption, (choices|item|itemset)+, (help|hint|alert|action|
extension)* -->
</selectOne>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**

common attributes defined in **8.12.1 Common Attributes**
selectUI - appearance override
selection - optional attribute determining whether free entry is allowed in the list.

8.10 selectMany

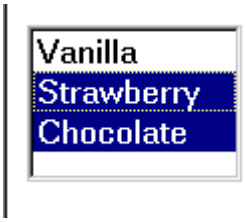
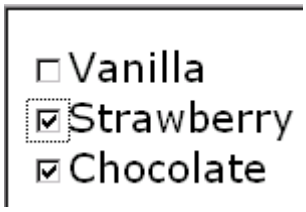
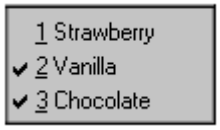
Description: This form control allows the user to make multiple selections from a set of choices.

Selecting Ice Cream Flavor

```
<selectMany ref="my:icecream/my:flavors">
  <caption>Flavours</caption>
  <choices>
    <item>
      <caption>Vanilla</caption>
      <value>v</value>
    </item>
    <item>
      <caption>Strawberry</caption>
      <value>s</value>
    </item>
    <item>
      <caption>Chocolate</caption>
      <value>c</value>
    </item>
  </choices>
</selectMany>
```

In the above example, more than one flavor can be selected.

A graphical browser might render form control `selectMany` as any of the following:

listbox	checkbox	radio	menu
		This configuration is not recommended.	

Data Binding Restrictions: any simpleContent capable of holding a sequence.

Note:

A limitation of the Schema list datatypes is that whitespace characters in the storage values (the `value="..."` attribute of the `item` element) are always interpreted as separators between individual data values. Therefore, authors

should avoid using whitespace characters within storage values with list simpleContent.

Incorrect Type Declaration

```
<item>
  <value>United States of America</value>
  ...
</item>
```

When selected, this item would introduce not one but four additional selection values: "America", "of", "States", and "United".

Implementation Hints: An accessibility aid might allow the user to browse through the available choices and leverage the grouping of choices in the markup to provide enhanced navigation through long lists of choices.

XML Representation: <`selectMany`>

```
<selectMany
  (single node binding attributes)
  (common attributes)
  selectUI = ("radio" | "checkbox" | "menu" | "listbox")
>
  <!-- caption, (choices|item|itemset)+, (help|hint|alert|action|
extension)* -->
</selectMany>
```

(single node binding attributes) - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
common attributes defined in **8.12.1 Common Attributes**
selectUI - appearance override

8.11 Common Markup for selection controls

8.11.1 choices

This element is used within selection form controls to group available choices. This provides the same functionality as element `optgroup` in HTML.

XML Representation: <`choices`>

```
<choices>
  <!-- caption?, (choices|item|itemset)+ -->
</choices>
```

8.11.2 item

This element specifies the storage value and caption to represent an item in a list. It is found within elements `selectOne` and `selectMany`, or grouped in element `choices`.

XML Representation: <`item`>

```
<item>
  <!-- caption, value, (help|hint|alert|action|extension)* -->
</item>
```

id = xsd:ID - optional unique identifier.

8.11.3 itemset

Element `itemset` allows the creation of dynamic selections within controls `selectOne` and `selectMany`, where the available choices are determined at run-time. The node-set that holds the available choices is specified via attribute `nodeset`. Child elements `caption` and `value` indirectly specify the caption and storage values. Notice that the run-time effect of `itemset` is the same as using element `choices` to statically author the available choices.

XML Representation: <`itemset`>

```
<itemset
  (node-set binding attributes)
>
  <!-- caption, value, (help|hint|alert|action|extension)* -->
</itemset>
```

node-set binding attributes - required node-set selector that specifies the node-set holding the available choices.

The following example shows element `itemset` within control `selectMany` to specify a dynamic list of ice cream flavors:

Dynamic Choice Of Ice Cream Flavors

```
<model id="cone">
  <instance>
    <my:icecream>
      <my:flavours/>
    </my:icecream>
  </instance>
</model>
<model id="flavours">
  <instance>
    <my:flavours>
      <my:flavour type="v">
        <my:description>Vanilla</my:description>
      </my:flavour>
      <my:flavour type="s">
        <my:description>Strawberry</my:description>
      </my:flavour>
      <my:flavour type="c">
        <my:description>Chocolate</my:description>
      </my:flavour>
    </my:flavours>
  </instance>
</model>
<!-- user interaction markup -->
<selectMany model="cone" ref="my:icecream/my:flavours">
```

```
  <caption>Flavors</caption>
  <itemset model="flavours" nodeset="my:flavours/my:flavour">
    <caption ref="my:description"/>
    <value ref="@type"/>
  </itemset>
</selectMany>
```

8.11.4 value

This element provides a storage value to be used when an `item` is selected.

Data Binding Restriction: All lexical values must be valid according to the datatype bound to the selection control.

XML Representation: <`value`>

```
<value
  (single node binding attributes)
>
  <!-- ##any -->
</value>
```

single node binding attributes - optional binding selector that specifies a location from where the storage value is to be fetched.

If inline content and a `ref` attribute are both specified, the `ref` attribute is used.

8.12 Common Markup

The preceding form control definitions make reference to child elements and attributes that are common to several of the form controls. This section defines these common markup components.

8.12.1 Common Attributes

The following attributes are common to many user-interface related XForms elements.

XML Representation: Common Attributes

```
xml:lang = xsd:language
class = space separated list of classes
navIndex = xsd:nonNegativeInteger : 0
accessKey = xsd:token
```

xml:lang - Optional standard XML attribute to specify a human language for this element.
class - Optional selector for a style rule.
navIndex - Optional attribute is a non-negative integer in the range of 0-32767 used to define the navigation sequence. This gives the author control over the sequence in which form controls are traversed. The default navigation order is specified in the chapter **4 Processing Model**.
accessKey - Optional attribute defines a shortcut for moving the input focus

directly to a particular form control. The value of this is typically a single character which when pressed together with a platform specific modifier key (e.g., the *alt* key) results in the focus being set to this form control.

8.12.2 Single Node Binding Attributes

The following attributes define a binding between a form control and an instance data node.

XML Representation: Single Node Binding Attributes

```
ref = binding-expression
model = xsd:IDREF
bind = xsd:IDREF
```

ref - Binding expression. Details in the chapter **6 Constraints**. The first-node rule applies to the nodeset selected here.
model - Optional instance data selector. Details in the section **6.4.3 Binding References**.
bind - Optional reference to a bind element

It is an error if the `model` idref value refers to an id not on a `model` element, or if the `bind` idref value refers to an id not on a `bind` element.

8.12.3 Nodeset Binding Attributes

The following attributes define a binding between a form control and a node-set returned by the XPath expression.

XML Representation: Nodeset Binding Attributes

```
nodeset = binding-expression
model = xsd:IDREF
bind = xsd:IDREF
```

nodeset - Binding expression. Details in the chapter **6 Constraints**.
model - Optional instance data selector. Details in the chapter **6 Constraints**.
bind - Optional reference to a bind element

It is an error if the `model` idref value refers to an id not on a `model` element, or if the `bind` idref value refers to an id not on a `bind` element.

8.12.4 Common Child Elements

The child elements detailed below provide the ability to attach metadata to form controls.

Instead of supplying such metadata e.g., the label for a form control as inline content of the contained element `caption`, the metadata can be pointed to by using a simple XLink attribute `xlink:href` on these elements. Notice that systematic use of this feature can be exploited in internationalizing XForms user interfaces by:

- Factoring all human readable messages to a separate resource XML file.
- Using URIs into this XML resource bundle within individual caption elements
- Finally, an XForms processor can use content negotiation to obtain the appropriate XML resource bundle, e.g., based on the `accept-language` headers from the client, to serve up the user interface with messages localized to the client's locale.

8.12.4.1 caption

The required element `caption` labels the containing form control with a descriptive label. Additionally, the caption makes it possible for someone who can't see the form control to obtain a short description while navigating between form controls.

XML Representation: <`caption`>

```
<caption
  (common attributes)
  (single node binding attributes)
  xlink:href = xsd:anyURI
>
  <!-- ##any -->
</caption>
```

common attributes - defined in **8.12.1 Common Attributes**
single node binding attributes - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
xlink:href = xsd:anyURI - link to external caption

The caption specified can exist in instance data, in a remote document, or as inline text. If multiple captions are specified in this element, the order of preference is: `ref`, `xlink:href`, inline.

An accessibility aid would typically speak the metadata encapsulated here when the containing form control gets focus.

8.12.4.2 help

The optional element `help` provides a convenient way to attach help information to a form control. This is equivalent to a `xforms:help` event handler that responds with a `<message type="modeless">`.

XML Representation: <`help`>

```
<help
  (common attributes)
  (single node binding attributes)
  xlink:href = xsd:anyURI
```

```
>
  <!-- ##any -->
</help>
```

(common attributes) - defined in **8.12.1 Common Attributes**
single node binding attributes - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
xlink:href = xsd:anyURI - link to external help

The message specified can exist in instance data, in a remote document, or as inline text. If multiple captions are specified in this element, the order of precedence is: `ref`, `xlink:href`, inline.

8.12.4.3 hint

The optional element `hint` provides a convenient way to attach hint information to a form control. This is equivalent to a `xforms:hint` event handler that responds with a `<message type="ephemeral">`.

XML Representation: <`hint`>

```
<hint
  (common attributes)
  (single node binding attributes)
  xlink:href = xsd:anyURI
>
  <!-- ##any -->
</hint>
```

(common attributes) - defined in **8.12.1 Common Attributes**
single node binding attributes - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
xlink:href = xsd:anyURI - link to external hint

The message specified can exist in instance data, in a remote document, or as inline text. If multiple captions are specified in this element, the order of precedence is: `ref`, `xlink:href`, inline.

8.12.4.4 alert

The optional element `alert` provides a convenient way to attach alert or error information to a form control. This is equivalent to a `xforms:alert` event handler that responds with a `<message type="modal">`.

XML Representation: <`alert`>

```
<alert
  (common attributes)
  (single node binding attributes)
  xlink:href = xsd:anyURI
>
  <!-- ##any -->
</alert>
```

(common attributes) - defined in **8.12.1 Common Attributes**
single node binding attributes - Selection of instance data node, defined in **8.12.2 Single Node Binding Attributes**
xlink:href = xsd:anyURI - link to external alert

The message specified can exist at in instance data, in a remote document, or as inline text. If multiple captions are specified in this element, the order of precedence is: `ref`, `xlink:href`, inline.

8.12.4.5 extension

Optional element `extension` is a container for application-specific extension elements from any namespace other than the XForms namespace. This specification does not define the processing of this element.

XML Representation: <`extension`>

```
<extension>
  <!-- ##other -->
</extension>
```

For example, RDF metadata could be attached to an individual form control as follows:

```
<input ref="dataset/user/email" id="email-input">
  <caption>Enter your email address</caption>
  <extension>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about="#email-input">
      <my:addressBook>personal</my:addressBook>
      </rdf:Description>
    </rdf:RDF>
  </extension>
</input>
```

Q 3) Write a note Comments in PHP.
Ans:- PHP Comment
Comments play very important role at time of development. A Comment in PHP code is short note that PHP engine ignores. PHP comment is used to provide the short description of our codeto the other developer or for own understanding. Some important about PHP comment
1. Comments are ignored by the browser, but useful for usand as well as other programmers.
2. It is good programming practice to use the comment.
3. The Comment does not show on the browser.PHP supports two-way of commenting
1. Single line comment
2. Multiple line comment All Rights Reserved
Single line comment
1)Single line comment has two types
Using "#"
Using "//"
PHP Comment Syntax
// this is comment line

```
# this is comment line
PHP Single Line Comment Example
<?php
echo"hello welcome to php". "<br>";
//Comment line
echo "Hello". "<br>";
// this is my Second output statement
echo date("Y/m/d") . "<br>";
# Display current_date
echo date("Y.m.d") . "<br>";
// in different format
echo date("Y-m-d");
?>· All Rights Reserved
```

OUTPUT

```
hello welcome to php
Hello
2015/09/20
2015.09.20
2015-09-20
```

Explanation

In the above example, line number 4, 6, 8 and 10 are comments line that will not display on the Web Browser.

Multiple line comment example

```
<?php
/*echo"hello welcome to php". "<br>";
echo "Hello". "<br>";
echo date("Y/m/d") . "<br>";
echo date("Y.m.d") . "<br>"; */
echo date("Y-m-d");
echo "Only two echo will be executed";
?>Reserved
```

OUTPUT

```
2015-09-20
Only two echo will be executed
```

Explanation

You can make comment, any statement, function, variable etc by using /* */ or // or #.

In the above example from line 3 to 9 are not executed and rest of the statements will execute properly.

Q 4) Explain different Debugging steps in PHP

Ans:-

Nobody enjoys the process of debugging their code. If you want to build killer web apps though, it’s vital that you understand the process thoroughly.

This article breaks down the fundamentals of debugging in PHP, helps you understand PHP's error messages and introduces you to some useful tools to help make the process a little less painful.

Doing your Ground Work

It is important that you configure PHP correctly and write your code in such a way that it produces meaningful errors at the right time. For example, it is generally good practice to turn on a verbose level of error reporting on your development platform. This probably isn't such a great idea, however, on your production server(s). In a live environment you neither want to confuse a genuine user or give malicious users too much information about the inner-workings of your site.

So, with that in mind lets talk about the all too common "I'm getting no error message" issue. This is normally caused by a syntax error on a platform where the developer has not done their ground work properly. First, you should turn display_errors on. This can be done either in your php.ini file or at the head of your code like this:

```
<?php
ini_set('display_errors', 'On');
```

Tip: In these code examples I omit the closing (?>) PHP tag. It is generally considered good practice to do so in files which contain only PHP code in order to avoid accidental injection of white space and the all too common "headers already sent" error.

Next, you will need to set an error reporting level. As default PHP 4 and 5 do not show PHP notices which can be important in debugging your code (more on that shortly). Notices are generated by PHP whether they are displayed or not, so deploying code with twenty notices being generated has an impact upon the overhead of your site. So, to ensure notices are displayed, set your error reporting level either in your php.ini or amend your runtime code to look like this:

```
<?php
ini_set('display_errors', 'On');
error_reporting(E_ALL);
```

Tip: E_ALL is a constant so don't make the mistake of enclosing it in quotation marks.

With PHP 5 it's also a good idea to turn on the E_STRICT level of error reporting. E_STRICT is useful for ensuring you're coding using the best possible standards. For example E_STRICT helps by warning you that you're using a deprecated function. Here's how to enable it at runtime:

```
<?php
ini_set('display_errors', 'On');
error_reporting(E_ALL | E_STRICT);
```

It is also worth mentioning that on your development platform it is often a good idea to make these changes in your php.ini file rather than at the runtime. This is because if you experience a syntax error with these options set in your code and not in the php.ini you may, depending on your set up, be presented with a blank page. Likewise, it is worth noting that if you're setting

these values in your code, a conditional statement might be a good idea to avoid these settings accidentally being deployed to a live environment.

What Type of Error am I Looking at?

As with most languages, PHP's errors may appear somewhat esoteric, but there are in fact only four key types of error that you need to remember:

1. Syntax Errors

Syntactical errors or parse errors are generally caused by a typo in your code. For example a missing semicolon, quotation mark, brace or parentheses. When you encounter a syntax error you will receive an error similar to this:

```
Parse error: syntax error, unexpected T_ECHO in /Document/Root/example.php on line 6
```

In this instance it is important that you check the line above the line quoted in the error (in this case line 5) because while PHP has encountered something unexpected on line 6, it is common that it is a typo on the line above causing the error. Here's an example:

```
<?php
ini_set('display_errors', 'On');
error_reporting(E_ALL);

$sSiteName = "Treehouse Blog"
echo $sSiteName;
```

In this example I have omitted the semi-colon from line 5, however, PHP has reported an error occurred on line 6. Looking one line above you can spot and rectify the problem.

Tip: In this example I am using Hungarian Notation. Adopting this coding standard can aid with debugging code while working collaboratively or on a piece of code you wrote some time ago. The leading letter denoting the variable type means that determining a variable type is very quick and simple. This can aid in spotting irregularities which can also help highlight any potential logic errors.

2. Warnings

Warnings aren't deal breakers like syntax errors. PHP can cope with a warning, however, it knows that you probably made a mistake somewhere and is notifying you about it. Warnings often appear for the following reasons:

Headers already sent. Try checking for white space at the head of your code or in files you're including.

You're passing an incorrect number of parameters to a function.

Incorrect path names when including files.

3. Notices

Notices aren't going to halt the execution of your code either, but they can be very important in tracking down a pesky bug. Often you'll find that code that's working perfectly happily in a production environment starts throwing out notices when you set error_reporting to E_ALL.

A common notice you'll encounter during development is:

>Notice: Undefined index: FullName in /Document/Root/views/userdetails.phtml on line 55

This information can be extremely useful in debugging your application. Say you've done a simple database query and pulled a row of user data from a table. For presentation in your view you've assigned the details to an array called $aUserDetails. However, when you echo $aUserDetails['FirstName'] on line 55 there's no output and PHP throws the notice above. In this instance the notice you receive can really help.

PHP has helpfully told us that the FirstName key is undefined so we know that this isn't a case of the database record being NULL. However, perhaps we should check our SQL statement to ensure we've actually retrieved the user's first name from the database. In this case, the notice has helped us rule out a potential issue which has in turn steered us towards the likely source of our problem. Without the notice our likely first stop would have been the database record, followed by tracing back through our logic to eventually find our omission in the SQL.

4. Fatal Errors

Fatal Errors sound the most painful of the four but are in fact often the easiest to resolve. What it means, in short, is that PHP understands what you've asked it to do but can't carry out the request. Your syntax is correct, you're speaking its language but PHP doesn't have what it needs to comply. The most common fatal error is an undefined class or function and the error generated normally points straight to the root of the problem:

Fatal error: Call to undefined function create() in /Document/Root/example.php on line 23

Using var_dump() to Aid Your Debugging

var_dump() is a native PHP function which displays structured, humanly readable, information about one (or more) expressions. This is particularly useful when dealing with arrays and objects as var_dump() displays their structure recursively giving you the best possible picture of what's going on. Here's an example of how to use var_dump() in context:

Below I have created an array of scores achieved by users but one value in my array is subtly distinct from the others, var_dump() can help us discover that distinction.

```
<?php ini_set('display_errors', 'On'); error_reporting(E_ALL); $aUserScores = array('Ben' =>
7,'Linda' => 4,'Tony' => 5,'Alice' => '9'); echo '<pre>'; var_dump($aUserScores); echo '</pre>';
```

Tip: Wrap var_dump() in <pre> tags to aid readability.

The output from var_dump() will look like this:

```
array(4) { ["Ben"]=> int(7) ["Linda"]=> int(4) ["Tony"]=> int(5) ["Alice"]=> string(1) "9" }
```

As you can see var_dump tells us that $aUserScores is an array with four key/value pairs. Ben, Linda, and Tony all have their values (or scores) stored as integers. However, Alice is showing up as a string of one character in length.

If we return to my code, we can see that I have mistakenly wrapped Alice's score of 9 in quotation marks causing PHP to interpret it as a string. Now, this mistake won't have a massively adverse effect, however, it does demonstrate the power of var_dump() in helping us get better visibility of our arrays and objects.

While this is a very basic example of how var_dump() functions it can similarly be used to inspect large multi-dimensional arrays or objects. It is particularly useful in discovering if you have the correct data returned from a database query or when exploring a JSON response from say, Twitter:

```
<?php
ini_set('display_errors', 'On');
error_reporting(E_ALL);

$sJsonUrl = 'http://search.twitter.com/trends.json';

$sJson = file_get_contents($sJsonUrl,0,NULL,NULL);
$oTrends = json_decode($sJson);

echo '<pre>';
var_dump($oTrends);
echo '</pre>';
```

Useful Tools to Consider when Debugging

Finally, I want to point out a couple of useful tools that I've used to help me in the debugging process. I won't go into detail about installing and configuring these extensions and add-ons, but I wanted to mention them because they can really make our lives easier.

Xdebug

Xdebug is a PHP extension that aims to lend a helping hand in the process of debugging your applications. Xdebug offers features like:

Automatic stack trace upon error

Function call logging

Display features such as enhanced var_dump() output and code coverage information.

Xdebug is highly configurable and adaptable to a variety of situations. For example, stack traces (which are extremely useful for monitoring what your application is doing and when) can be configured to four different levels of detail. This means that you can adjust the sensitivity of Xdebug's output helping you to get granular information about your app's activity.

Stack traces show you where errors occur, allow you to trace function calls and detail the originating line numbers of these events. All of which is fantastic information for debugging your code.

Tip: As default Xdebug limits var_dump() output to three levels of recursion. You may want to change this in your xdebug.ini file by setting the xdebug.var_display_max_depth to equal a number that makes sense for your needs.

Check out Xdebug's installation guide to get started.

FirePHP

For all you FireBug fans out there, FirePHP is a really useful little PHP library and Firefox add-on that can really help with AJAX development.

Essentially FirePHP enables you to log debug information to the Firebug console using a simple method call like so:

```
<?php
$sSql = 'SELECT * FROM tbl';
FB::log('SQL query: ' . $sSql);
```

In an instance where I'm making an AJAX search request, for example, it might be useful to pass back the SQL string my code is constructing in order that I can ensure my code is behaving correctly. All data logged to the Firebug console is sent via response headers and therefore doesn't affect the page being rendered by the browser.

Warning: As with all debug information, this kind of data shouldn't be for public consumption. The downside of having to add the FirePHP method calls into your PHP is that before you go live you will either have to strip all these calls out or set up an environment based conditional statement which establishes whether or not to include the debug code.

You can install the Firefox add-on at FirePHP's website and also grab the PHP libsthere too. Oh, and don't forget if you haven't already installed FireBug, you'll need that too.

In Conclusion…

Hopefully, during the course of this article you have learned how to do your groundwork by preparing PHP for the debugging process; recognize and deal with the four key PHP error types and use var_dump() to your advantage. Likewise, I hope that you will find Xdebug and FirePHP useful and that they will make your life easier during your development cycle.

As I've already mentioned, and I really can't say this enough, always remember to remove or suppress your debug output when you put your sites into production, after all, there's nothing worse than all your users being able to read about your errors in excruciating detail.

Got a great debugging tip to share? Do you use a great little PHP extension that makes your bug trapping life easier? Please tell us about them in comments below!

Q 5) Different between Server Page and Client Page.

Ans:- Client-side Environment

The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the users computer over the internet and run directly in the browser.

The scripting language needs to be **enabled** on the client computer. Sometimes if a user is conscious of **security risks** they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.

Server-side Environment

The **server-side environment** that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server.

This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

Background

Web development is all about communication. In this case, communication between two (2) parties, over the HTTP protocol:

- The Server - This party is responsible for serving pages.
- The Client - This party *requests* pages from the Server, and displays them to the user. In most cases, the client is a web browser.
    - The User - The user *uses* the Client in order to surf the web, fill in forms, watch videos online, etc.

Each side's programming, refers to code which runs at the specific machine, the server's or the client's.

Basic Example

1. The User opens his web browser (the Client).
2. The User browses to http://google.com.
3. The Client (on the behalf of the User), sends a request to http://google.com (the Server), for their home page.
4. The Server then acknowledges the request, and replies the client with some meta-data (called *headers*), followed by the page's source.
5. The Client then receives the page's source, and *renders* it into a human viewable website.
6. The User types Stack Overflow into the search bar, and presses Enter
7. The Client submits that data to the Server.
8. The Server processes that data, and replies with a page matching the search results.
9. The Client, once again, renders that page for the User to view.

Programming

Server-side Programming

Server-side programming, is the general name for the kinds of programs which are run on the Server.

Uses

- Process user input.
- Display pages.
- Structure web applications.
- Interact with permanent storage (SQL, files).

Example Languages

- PHP
- Python
- ASP.Net in C#, C++, or Visual Basic.
- Nearly any language (C++, C#, Java). These were not designed specifically for the task, but are now often used for application-level web services.

Client-side programming

Much like the server-side, Client-side programming is the name for all of the programs which are run on the Client.

Uses

- Make interactive webpages.
- Make stuff happen dynamically on the web page.
- Interact with temporary storage, and local storage (Cookies, localStorage).
- Send requests to the server, and retrieve data from it.
- Provide a remote service for client-side applications, such as software registration, content delivery, or remote multi-player gaming.

Example languages

- JavaScript (primarily)
- HTML*
- CSS*
- Any language running on a client device that interacts with a remote service is a client-side language.

Q6) Explain Comparison operation in PHP.

Ans:- In PHP, comparison operators take simple values (numbers or strings) as arguments and evaluate to either TRUE or FALSE.

Here is a list of comparison operators.

Operator	Name	Example	Result
= =	Equal	$x == $y	TRUE if $x is exactly equal to $y
= = =	Identical	$x === $y	TRUE if $x is exactly equal to $y, and they are of the same type.

!=	Not equal	$x != $y	TRUE if $x is exactly not equal to $y.
<>	Not equal	$x <> $y	TRUE if $x is exactly 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 (left-hand argument) is strictly less than $y (right-hand argument).
>	Greater than	$x > $y	TRUE if $x (left hand argument) is strictly greater than $y (right hand argument).
<=	Less than or equal to	$x <= $y	TRUE if $x (left hand argument) is less than or equal to $y (right hand argument).
>=	Greater than or equal to	$x >= $y	TRUE if $x is greater than or equal to $y.

Q 7) Explain the syntax of PHP program. Write a program to find square of 2.
Ans:-
The rules that must be followed to write properly structured code.
PHP's syntax and semantics are similar to most other programming languages (C, Java, Perl) with the addition that all PHP code is contained with a tag, of sorts. All PHP code must be contained within the following
Syntax:-
<?php
or the shorthand PHP tag that requires shorthand support to be enabled
on your server...
?>
If you are writing PHP scripts and plan on distributing them, we suggest that you use the standard form (which includes the ?php) rather than the shorthand form. This will ensure that your scripts will work, even when running on other servers with different settings.
How to save PHP Pages If you have PHP inserted into your HTML and want the web browser to interpret it correctly, then you must save the file with a *.php* extension, instead of the standard *.html* extension. So be sure to check that you are saving your files correctly. Instead of *index.html*, it should be *index.php* if there is PHP code in the file. Below is an example of one of the easiest PHP and HTML page that you can create and still follow web standards.

If you save this file (e.g. helloworld.php) and place it on PHP enabled server and load it up in your web browser, then you should see "Hello World!" displayed. If not, please check that you followed our example correctly.We used the PHP command *echo* to write "Hello World!" and we will be talking in greater depth about how echo is special later on in this tutorial.

The Semicolon:-As you may or may not have noticed in the above example, there was a semicolon after the line of PHP code. The semicolon signifies the end of a PHP statement and should never be forgotten. For example, if we repeated our "Hello World!" code several times, then we would need to place a semicolon at the end of each statement.

White Space:-As with HTML, whitespace is ignored between PHP statements. This means it is OK to have one line of PHP code, then 20 lines of blank space before the next line of PHP code. You can also press tab to indent your code and the PHP interpreter will ignore those spaces as well. **Example:**-
<html>
<body>
<?php

```
echo(pow(2,2) . "<br>");
?>
</body>
</html>
```

Output:-

4

Part - C (Each question carries Five marks)

UNIT – II

1.Explain any four string function with example.

Ans:-1.**php**strtolower():-the strtolower()function returns string to lowercase letters.

This function binary safe.

Example:-

```
<php
    $str =”My name is Khan”;
     $str =”strtolower($str);
    Echo $str;
     ?>
```

2.phpstrtoupper():- The string strtoupper() function return string in uppercase to letter .This function is binary-safe. Returns string with all alphabetic charactersconverted to uppercase.

Example:- :-

```
<php
    $str =”My name is Khan”;
     $str =”strtoupper($str);
    Echo $str;
     ?>
```

3.phpUCfirst():-The uc function return string converting first character into appear. It does change the can of other change.

Example:- :-

```
<php
    $str =”My name is Khan”;
     $str =”UCfirst($str);
    Echo $str;
     ?>
```

4.phplcfirst():-The lcfirst() function return string character into lowercase. It doesn’t change the case of other character.

Example:- :-

```
<php
    $str =”My name is Khan”;
     $str =”lcfirst($str);
    Echo $str;
     ?>
```

2.What is function ? explain user define function with example.

Ans:-Phpuser define functions, Besides the built - in php function .We can create our own functions. A functions is a block of statements that can be used repeatedly in a program . A function will not execute immediately when a page loads. A functions will be executed by a call to the function.

Example:-<?php

```
function function1()
   {
   Function function2()
   {
   Echo “Good Morning<br>”;
    }
   }
function1();
Function2();
    ?>
```

3.What is array? Explain associative array with example.

Ans:-Php array is an ordered map contents value on the bases of key. It is use to hold multiple value of similar type in a singlr variable.

Associative array:-we can associate name with each array element in phpusin (□).In the product array,We allowed php to give each item the default index. This meant the first item,we added became 0,The second item 1, and so on.php also supports associative array,In an associative array,we can associate any key or index we want with each value.

Example:-<?php

```
$salary=array(“abc”□”5000”,
        ”pqr”□”15000”,
            “xyz”□”10,000”);
  Echo $salary are :-;
  $salary[abc],$salary[pqr],$salary[xyz];
   ?>
```

4.Write a notes on switch statement in php.

Ans:-The switch statement is used to perform different action based on different conditions.Use the many blocks of code to be executed .The switch statement is similar to if statements on the same expression.In many occasions, you may want to compare the same variable with many different values, and execute a different piece of code depending on which value it equals to.

5.Explain if….else statement .write a program to print all even numbers from 1 to100.

Ans:-php conditional statment .Very often when you write code, you want to perform different action for diffrentconditions.you can use conditional statments your code to do this if ....else statment execute some code if a condition is true and another code if condition is false.

Example:-

```
<?php
$num=100;
If($num%2=0)
{
Echo “$num is even number”;
}
Else
{
Echo ”$num is not even number”;
}
?>
```

6.Explain for loop with example.
Ans:-For loop can be use to travels set of code for the number of tags. The for loop repeats a block of code until a certain condition is met. It is typically used to execute a block of code certain number of time.
Example:-

```
<?php
For($x=0;$x<=10;$x++)
Echo “The number is :$x<br>”;
?>
```

7.Explain array_ splice () in php with example.
Ans:-The array _splice () function remove selected elements from an array and replaces. It with new elements. The function also return an array with the removed element.
Example:-

```
<?php
$a1=array(“a”□”red”, “b”□”green”,”c”□”blue”,”d”□”yellow”);
$a2=array(“a”□”purple”,”b”□”orange”);
array_splice($a1,0,2,$a2);
print_r($a1);
?>
```

8.List and explain all sort function in php.
Ans:-1.sort():-The sort () function sorts an indexed array in ascending order.
2.rsort():-sort array in descending order.
3.asort():-sort associative arrays in ascending order, according to the value.
4.ksort():-sort associative arrays in ascending order, according to the key.
5.arsort():-sort associative arrays in descending order , according to the value.
6.krsort():-sort associative arrays in descending order , according to the key.

9.Explainisset () function with example.
Ans:-The isset () function is used to check whether a variable is set oe not. If a variable is already unset with unset() function , it will no longer be set. The isset () function return false if testing variable contains a NULL.
Example:-

```
<?php
$var1=’test’;
Var_dump(isset($var1)
?>
```

10.Explian any three super global variable with example.
Ans:-1.php $_GET:-php $_GET can also be used to collect from data after submitting an html form with method =”get”. $_GET can also collect data sent in the URL.
2.php $_POST:- php $_POST is widely used to collect form data after submitting an html form with method=”post”. $_POST is also widely used to pass variables
3.php $_REQUEST:-php $_REQUEST is used to collect data after submitting an html form.

4.php $_SERVER :-$_SERVER is a php super global variable which hold information about headers, path, and script locations.
5php $_GLOBALS:-$_GLOBALS is a php super global variable which is used to access global variables from anywhere in the php script (also from within functions or methods).php stores all global variables in an array called $_ GLOBALS [index]. The index holds the name of the variable.

Part - C (Each question carries Five marks)
UNIT – III

1**. Explain date function with five formatting character.**
Answer:

The PHP date() function is used to format a date and/or a time.

The PHP Date() Function

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

`date(`*`format,timestamp`*`)`

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

Get a Simple Date

The required *format* parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)

- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

The example below formats today's date in three different ways:

Example

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

PHP Tip - Automatic Copyright Year

Use the date() function to automatically update the copyright year on your website:

Example

```
&copy; 2010-<?php echo date("Y");?>
```

Get a Simple Time

Here are some characters that are commonly used for times:

- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:

Example

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

Note that the PHP date() function will return the current date/time of the server!

Get Your Time Zone

If the time you got back from the code is not the right time, it's probably because your server is in another country or set up for a different timezone.

So, if you need the time to be correct according to a specific location, you can set a timezone to use.

The example below sets the timezone to "America/New_York", then outputs the current time in the specified format:

Example

```php
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

Create a Date With PHP mktime()

The optional *timestamp* parameter in the date() function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used (as shown in the examples above).

The mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax

```
mktime(hour,minute,second,month,day,year)
```

The example below creates a date and time from a number of parameters in the mktime() function:

Example

```php
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

Create a Date From a String With PHP strtotime()

The PHP strtotime() function is used to convert a human readable string to a Unix time.

Syntax

```
strtotime(time,now)
```

The example below creates a date and time from the strtotime() function:

Example

```
<?php
$d=strtotime("10:30pm April 15 2014");
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

PHP is quite clever about converting a string to a date, so you can put in various values:

Example

```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

However, strtotime() is not perfect, so remember to check the strings you put in there.

More Date Examples

The example below outputs the dates for the next six Saturdays:

Example

```
<?php
$startdate = strtotime("Saturday");
$enddate = strtotime("+6 weeks", $startdate);

while ($startdate < $enddate) {
  echo date("M d", $startdate) . "<br>";
  $startdate = strtotime("+1 week", $startdate);
}
?>
```

The example below outputs the number of days until 4th of July:

Example

```
<?php
$d1=strtotime("July 04");
```

```
$d2=ceil(($d1-time())/60/60/24);
echo "There are " . $d2 ." days until 4th of July.";
?>
```

2. What is include statement explain with example.
Answer:

Include

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax

```
include 'filename';

or
```

```
require 'filename';
```

PHP include Examples

Example 1

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

To include the footer file in a page, use the include statement:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

3. What is require statement explain with example.

Answer:

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax

```
include 'filename';

or

require 'filename';
```

PHP include Examples

Example 1

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

To include the footer file in a page, use the include statement:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

Example 2

Assume we have a standard menu file called "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

All pages in the Web site should use this menu file. Here is how it can be done (we are using a element so that the menu easily can be styled with CSS later):

Example

```
<html>
<body>

<div class="menu">
<?php include 'menu.php';?>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>

</body>
</html>
```

Example 3

Assume we have a file called "vars.php", with some variables defined:

```
<?php
$color='red';
$car='BMW';
?>
```

Then, if we include the "vars.php" file, the variables can be used in the calling file:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'vars.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

PHP include vs. require

The require statement is also used to include a file into the PHP code.

However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>
```

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

Use require when the file is required by the application.

Use include when the file is not required and application should continue when file is not found.

4. What is the difference between include and require statement.
Answer:

Difference between include() and require() in PHP :

include()	requie()
include() function takes one argument , a path/name of the specified file you want to include.	require() function performs the similar functionality as include() function, takes one argument i.e. path/name of the file you want to include.
Syntax : include(File-name); **Example :** include (“header.php”)	**Syntax :** require(File-name); **Example :** require(“header.php”);
include() function **does not stop executio**n , just give warning and continues to execute the script in case of file name not found i..e missing/misnamed files .	require() function gives **fatal error** if it founds any problem like file not found or misnamed files and stops the execution once get fatal error.
include() function can be used in loops or **control structure**.	require() function can not be used in **loops** or control structures.
You can **return** a value from an included file **Ex:** $result = include ‘rtnvalue.php’;	File included as the result of a require statement cannot return a value.
You can use include when file is not required or not so important .	When file is required by the script, better to use require() instead of include().

5. Explain any two validation function with example.
Answer:

This and the next chapters show how to use PHP to validate form data.

PHP Form Validation

Think SECURITY when processing PHP forms!

These pages will show how to process PHP forms with security in mind. Proper validation of form data is important to protect your form from hackers and spammers!

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Big Note on PHP Form Security

The $_SERVER["PHP_SELF"] variable can be used by hackers!

If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

So far, so good.

However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

In this case, the above code will be translated to:

```
<form method="post"
action="test_form.php/"><script>alert('hacked')</script>
```

This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.

Be aware of that any JavaScript code can be added inside the <script> tag! A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

How To Avoid $_SERVER["PHP_SELF"] Exploits?

$_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post"
action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

The exploit attempt fails, and no harm is done!

Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example

```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

6. How to send email in PHP? Explain with example.
Answer:

Definition and Usage

The mail() function allows you to send emails directly from a script.

Syntax

mail(*to,subject,message,headers,parameters*);

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. **Windows note:** If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php $txt = str_replace("\n.", "\n..", $txt); ?>
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). **Note:** When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

Technical Details

Return Value:	Returns the hash value of the *address* parameter, or FALSE on failure. **Note:** Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The *parameter* parameter is disabled in safe mode PHP 4.0.5: The *parameter* parameter was added

Example

Send a simple email:

```
<?php
// the message
$msg = "First line of text\nSecond line of text";
```

```
// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```

Example 2

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

Example 3

Send an HTML email:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
</body>
</html>
```

";

// Always set content-type when sending HTML email
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

// More headers
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";

mail($to,$subject,$message,$headers);
?>

7. Explain manipulating HTTP header with example.

Answer:

Manipulating HTTP Headers Most interactions between a server and a Web browser (the client) take place over HTTP (Hypertext Transfer Protocol). This is why the addresses for Web pages begin with http://. But the Web server often needs to communicate with a browser in other ways, beyond just sending HTML, images, and the like. These additional communications can be accomplished using HTTP headers. There are dozens of uses for HTTP headers, all of which you can do using PHP's header() function. Here, you'll learn a very common use of the header() function: redirecting the user from one page to another. To redirect the user's browser with PHP, you send a location header: header('Location: page.php'); Normally, the header() function is followed by exit(), to cancel the execution of the script (because the browser has been redirected to another page): header('Location: page.php'); exit(); The most important thing to understand about using header() is that the function must be called before anything else is sent to the Web browser—otherwise, you'll see the all-too-common "headers already sent" error message (see A in the section "Output Buffering"). If your Web page receives any HTML or even blank space, the header() function won't work.

Fortunately, you learned about output buffering in the previous section. Because output buffering is turned on in the Web application, nothing is sent to the Web browser until the very last line of the footer script (when ob_end_flush() is called). By using this method, you can avoid the dreaded "headers already sent" error message. To practice redirection, you'll rewrite the login page to take the user to a welcome page upon successful login. To use the header() function: 1. Open login.php in your text editor or IDE (Script 8.8). 2. Delete the You are logged in… print statement (Script 8.13). Because the user is redirected to another page, there's no need to include this message.

Definition and Usage

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

It is important to notice that header() must be called before any actual output is sent (In PHP 4 and later, you can use output buffering to solve this problem):

<html>
<?php

```
// This results in an error.
// The output above is before the header() call
header('Location: http://www.example.com/');
?>
```

Syntax

header(string,replace,http_response_code)

Parameter	Description
string	Required. Specifies the header string to send
replace	Optional. Indicates whether the header should replace previous or add a second header. Default is TRUE (will replace). FALSE (allows multiple headers of the same type)
http_response_code	Optional. Forces the HTTP response code to the specified value (available in PHP 4.3 and higher)

Tips and Notes

Note: Since PHP 4.4 this function prevents more than one header to be sent at once. This is a protection against header injection attacks.

Example 1

Prevent page caching:

```
<?php
// Date in the past
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Cache-Control: no-cache");
header("Pragma: no-cache");
?>

<html>
<body>

...
...
```

Note: There are options that users may set to change the browser's default caching settings. By sending the headers above, you should override any of those settings and force the browser to not cache!

Example 2

Let the user be prompted to save a generated PDF file (Content-Disposition header is used to supply a recommended filename and force the browser to display the save dialog box):

```
<?php
header("Content-type:application/pdf");

// It will be called downloaded.pdf
header("Content-Disposition:attachment;filename='downloaded.pdf'");

// The PDF source is in original.pdf
readfile("original.pdf");
?>

<html>
<body>

...
...
```

8. Write note on cookies in PHP.

Answer:

A cookie is often used to identify a user.

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
```

```
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
```

```
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

9. Write a note on session in PHP.
Answer:

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.

What is a PHP Session?

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.

So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a database.

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
```

```
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

How does it work? How does it know it's me?

Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

10. Write a note on $_SERVER['REQUEST_METHOD'].
Answer:

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

PHP Global Variables - Superglobals

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES

- $_ENV
- $_COOKIE
- $_SESSION

This chapter will explain some of the superglobals, and the rest will be explained in later chapters.

PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

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

The example below shows how to use the super global variable $GLOBALS:

Example

```
<?php
$x = 75;
$y = 25;

function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!

PHP $_SERVER

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

Example

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

```
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

The following table lists the most important elements that can go inside $_SERVER:

Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page

$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

PHP $_REQUEST

PHP $_REQUEST is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
```

```
        echo $name;
    }
}
?>

</body>
</html>
```

PHP $_POST

PHP $_POST is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

PHP $_GET

PHP $_GET can also be used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

The example below shows the code in "test_get.php":

Example

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

11. What is function in PHP? Explain User defined function with example.

Answer:

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

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

Create a User Defined Function in PHP

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

Syntax

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

Note: A function name can start with a letter or underscore (not a number).

Tip: Give the function a name that reflects what the function does!

Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

Example

```
<?php
function writeMsg() {
  echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```
<?php
function familyName($fname) {
  echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

The following example has a function with two arguments ($fname and $year):

Example

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions - Returning values

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

Example

```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
```

```
echo "2 + 4 = " . sum(2, 4);
?>
```

Part - C (Each question carries Five marks)

UNIT – IV

. **1.Explain How to insert values from table in MySql.**

Ans:-After a database and a table have been created , we can start adding data in them .

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 NUL must not be quoted.

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

```
INSERT INTOtable_name
(column1, column2, column3……)
VALUES (value1, values2, values3…..)
```

To INSERT usingphpMyAdmin:

1.Log into your cPanel and click the phpMyAdmin icon.

2.In the left menu , first click your database name and then click the table to work with.

3. In the top menu , click “Insert”.

4. Type in a sample comment (refer to our screenshot below)and then click GO.

2.Explain different data types in MySql.

Ans:-A Data type specifies a particular type of data , such as integer ,floating – point , Boolean etc. a data type also specifies the possible values for that type , the operations that can be performed on that type and the way values of that type are stored .MySQL supports a number of SQL Standard data types in various categories. MySQL has numeric types ,the DATETIME ,DATE ,and TIMESTAMP Types and String Types.

1.MySQL Numeric Type:- MySQL supports all standard SQL numeric data types which include INTEGER , SMALLINT ,DECIMAL and NUMERIC . It also support also the approximate numeric data type.

1. Integer type:- SQL Standard integer types INTEGER ans SMALLINT are supported by MySQL . As an extension to the standard , MySQL also supports the integer type TINYINT , MEDIUMINT ,and BIGINT .

2. Floating –Point Types:-The FLOAT and DOUBLE TYPE represented approximate numeric data values .MySQL uses four bytes for single-precision values and eight byte for double – preprecision values.

3.Fixed –point Type:-Fixed –point data type are used to preserve exact precision , for example for currency data. In MySQL DECIMAL and NUMERIC types store exact numeric data values . MySQL 5.6 stores DECIMAL values in binary format.

4.L Bit value Type:-The BIT data type is used to store the bit – filed values. A type of BIT(N) enable storage. Of N-bit values .N can range from 1 to 64 . To specify bit values ,b'values' notation can be used. Value is a binary value written using zeros and once.
2.MySQL Date and Time Types:-The MySQL date and time data types are as Follow:-
1.DATE:-A date in YYYY-MM-DD format , between 1000-01-01 and 9999-12-31, for example, December 30th , 1973 would be stored as 1973-12-30
2.DATETIME:-A date and and time combination in YYYY-MM-DD HH:MM:SS format , between 9999-12-31 23:59:59. For example 3:30 in the afternoon on December 30th, 1933-12-30 15:30:00.
3.TIMESTAMP:-A timestamp between midnight, January 1st ,1970 and sometime in 2037 . This looks only without the hyphens between numbers; 3:30 in the afternoon on December 30th , 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
4.TIME:- Store the time in a HH:MM:SS format.
5.YEAR(M):-Store a year in a 2-digit or a 4 digit format . If length is specified as 2 YEAR can be between 1970 to 2069 .If the length is specified as 4, then YEAR can be 1901 to 2155 .The default value is 4.
3.String Type:-Although the numeric and date type are fun, most data you'll store will be in a string format. This list describes the common string datatype in MySQL.
1.CHAR(M):-A fixed- length string between 1 and 255 characters in length , right-padded with space to the specified length when stored. Defining a length is not required , but the default is 1.
2.VARCHAR(M):-A variable-length string between 1 and 255 characters in length. For example, VHARCHAR(25),you must define a length when creating a VHARCHAR field.

3.Explain how to delete values from table in mysql.
Ans:-

Delete Data From a MySQL Table Using MySQLi and PDO

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

Example (MySQLi Object-oriented)

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

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Procedural)

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

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

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";
```

```php
if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (PDO)

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

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to delete a record
    $sql = "DELETE FROM MyGuests WHERE id=3";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Record deleted successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

After the record is deleted, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

DELETE Syntax

```
DELETE FROM table_name
WHERE condition;
```

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SQL DELETE Example

The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

Example

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

The "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM *table_name*;

or:

DELETE * FROM *table_name*;

4.Explain REVOKE command in MySQL.

Ans:-It is used when the privilege being granted on a function in MySQL. Object. The name of the database object the you are granting privileges for. In the case of granting EXECUTE privileges on a function or procedure , this would be the function name or the procedure name. The REVOKE command removes user access right or privileges to the database objects.It can be used for removing the privileges assigned via its counterpart GRANT , in either the global , database or table scope.The REVOKE statement enables system administration to revoke privileges from Mysql account. When the read_only system variables in enabled , REVOKE requires the super privilege in addition to any order required privileges describe.

5.Explain UPDATE command in MySQL.

Ans:-An SQL UPDATE statement changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition The UPDATE statement has the following Form : UPDATE table_name. SET coiumn_name=value [column_name=value…]. For the UPDATE to be successful , the user must have data manipulation privileges on the table or column and the update value must not conflict with all the applicable constraints .The UPDATE statements updates column of existing rows in the name table with new values. The set clauses indicated which columns to modify and and the value they should be given . Each value can be given an expression or the keyword default to set a column explicitly .to its default value.

6.Explainmysql_fetch _array () function with example.

Ans:- The mysql_fetch_array () is used to retrieve a row of data as an array from a My SQL result handle. Refers to the resource return by the valid mysql query .The type of the result is an array. An array containing one row of data , or FALSE if there are no more rows. My_SQL_fetch_array returns the first row in a MySQL recourses in for form of an associative array.The column of the MySQL result can be accessedby using the column names of the table .
Example:-

```
<?php
$query = “SELECT *FROM example”;
$result = mysql_query($query) or die (mysql_error());
$row = mysql_featch_array($result) or die (mysql_error());the
Ecoh $row[‘name’]. “ _”. $row[‘age'];
?>
```

7.What is foreign key? Explain in detail with example.
Ans:-A FORIGN KEY IS used is a key use to link two tables together. A FORIGN KEY is a field in a one table that refers to the PRIMARY KEY in another table. The table containing the foreign key is called the child table , and the table containing the candidate key is called referenced or parent table. The FOREIGN KEY constraint used to prevent action that would destroys links between tables. The FOREGN KEY constraint also prevents invalid data from being . inserted into the foreign key column ,because it has to be one of the values contained in the table it points to.
Example:-

```
CREATE TABLE perents (
id INT NOT NULL ,
PRIMARY KEY (id)
) ENGINE =INNODB;
 CREATE TABLE chiled (
id INT ,
parent_id INT ,
INDEX par_ind (parent_id) ,
FORIGN KEY (parent_id)
REFERANCES parent_id)
ON DELETE CASECADE
) ENGINE=INNODB;
```

8.How to get information about database and table ? Explain .
Ans:-What if you forget the name of a database or table , or what the structure given table is .MySQL address this problem through information about the databases and tables it supports. You have previously seen SHOW DATABASES which lists the databased managed by server.To find out which database is currently selected , use the DATABASE ()Fuction. MySQL provides a SHOW statement that has several variant forms that display information about database and the table in them. SHOW helpful for keeping track of the contents of your databases and for reminding yourself about the structure of your table .

9.Explain REGEXP in mysql.
Ans:-MySQL regular Expression operator . The following operator are used in My SQL to perform regular expression operation . These are used in a WHERE clause similar to the well known and often use LIKE operator . REGEXP : the pattern matching operator for using regular expression. The pattern is supplied as an argument. If the pattern finds a match in the expression , the function returns 1, else it

returs 0. If either expression or pattern is NULL , the function returns NULL .The pattern can be extended regular expression , the syntax for which is discussed in Regular Expression Synatx . The pattern need not be a literal string .

10.Explain any three date function with example.
Ans:-1.CURRENT_DATE():-in MySQL , the CURRENT_DATE returns the current date in 'YYYY-MM-DD' format or YYYYMMDD format depending on whether numeric and string is used in function .
CURDATE() and CURRENT _DATE() are the synonym of CURRENT_DATE.
2.DATE_ADD ():- This functions perform date arithmetic . date is a DATETIME or DATE value specifying the string date . expression specify the interval value to be added or subtracted from the starting date.
3.DATE_FORMAT():-formats the date values according to the format string.The following specifies may be used in the format string. The % characters required before format specifiercharcters.

Part - C (Each question carries Ten marks)

UNIT – I

Unit-1
10-Marks

Q 1)Write a note on PHP, Explain How PHP Works in detail
Ans:- PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. Originally created by RasmusLerdorf in 1994, the PHP reference implementation is now produced by The PHP Group. PHP originally stood for *Personal Home Page*, but it now stands for the recursiveacronym*PHP: Hypertext Preprocessor*.PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalonegraphical applications. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge. The PHP language evolved without a written formal specification or standard until 2014, leaving the canonical PHP interpreter as a *de facto* standard. Since 2014 work has gone on to create a formal PHP specification.
How to work PHP:-

The PHP software works with the *web server,* which is the software that delivers web pages to the world. When you type a URL into your web browser's address bar, you're sending a message to the web server at that URL, asking it to send you an HTML file. The web server responds by sending the requested file. Your browser reads the HTML file and displays the web page.You also request a file from the web server when you click a link in a web page. In addition, the web server processes a file when you click a web page button that submits a form.

This process is essentially the same when PHP is installed. You request a file, the web server happens to be running PHP, and it sends HTML back to the browser, thanks to the programming in PHP.More specifically, when PHP is installed, the web server is configured to expect certain file extensions to contain PHP language statements. Often the extension is .php or .phtml, but any extension can be used. When the web server gets a request for a file with the designated extension, it sends the HTML statements as is, but PHP statements are processed by the PHP software before they're sent to the requester.When PHP language statements are processed, only the output, or anything printed to the screen is sent by the web server to the web browser. The PHP language statements, those that don't produce any output to the screen, aren't included in the output sent to the browser, so the PHP code is not normally seen by the user.
For instance, in this simple PHP statement, <?php is the PHP opening tag, and ?> is the closing tag.

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

Here, echo is a PHP instruction that tells PHP to output the upcoming text. The PHP software processes the PHP statement and outputs the following:

```
<p>Hello World</p>
```

That regular HTML statement is delivered to the user's browser. The browser interprets the statement as HTML code and displays a web page with one paragraph — Hello World. The PHP statement isn't delivered to the browser, so the user never sees any PHP statements. PHP and the web server must work closely together.PHP isn't integrated with all web servers but does work with many of the popular web servers. PHP works well with the Apache web server. PHP also works with Microsoft Internet Infor

Q 2)Explain in detail the role of Web Browser in detail.
Ans:-
Q 3)Explain GET and POST method in detail.
Ans:- There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

name1=value1&name2=value2&name3=value3
Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ?character.

http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.

- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides $_GET associative array to access all the sent information using GET method.

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not 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 header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides $_POST associative array to access all the sent information using POST method.

Q 4)What is Variable ? Explain different type of variables.

Ans:-Variables are used to store information to be referenced and manipulated in a computer program. They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves. It is helpful to think of variables as containers that hold information. Their sole purpose is to label and store data in memory. This data can then be used throughout your program.

Type of variable:-

1) ConstantsVariable
2) Global variables
3) Class variables
4) Instance variables
5) local variables.

1)Constants are declared by capitalizing every letter in the variable's name. They are used for storing data that never needs to change. While most programming languages do not allow you to change the value assigned to a constant, Ruby does. It will however throw a warning letting you know that there was a previous definition for that variable. Just because you can, doesn't mean you should change the value. In fact, you should not. Constants cannot be declared in method definitions, and are available throughout your application's scopes.

Example

MY_CONSTANT='I am available throughout your app.'

2)Global variables are declared by starting the variable name with the dollar sign ($). These variables are available throughout your entire app, overriding all scope boundaries. Rubyists tend to stay away from global variables as there can be unexpected complications when using them.

Example

$var='I am also available throughout your app.'

3)Class variables are declared by starting the variable name with two @ signs. These variables are accessible by instances of your class, as well as the class itself. When you need to declare a variable that is related to a class, but each instance of that class does not need its own value for this variable, you use a class variable. Class variables must be initialized at the class level, outside of any method definitions. They can then be altered using class or instance method definitions.

Example

@@instances=0

4)Instance variables are declared by starting the variable name with one @ sign. These variables are available throughout the current instance of the parent class. Instance variables can cross some scope boundaries, but not all of them. You will learn more about this when you get to OOP topics, and should not use instance variables until you know more about them.

Example

@var='I am available throughout the current instance of this class.'

5)Local variables are the most common variables you will come across and obey all scope boundaries. These variables are declared by starting the variable name with neither $ nor @, as well as not capitalizing the entire variable name.

Example

var='I must be passed around to cross scope boundaries.'

Q 5)Explain super Global Variable in PHP?
Ans:- PHP Global Variables - Superglobals

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

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

PHP $_SERVER

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

PHP $_REQUEST

PHP $_REQUEST is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

PHP $_POST

PHP $_POST is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

PHP $_GET can also be used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

The example below shows the code in "test_get.php":

Q 6)What is manually sending data to a form? Explain with example.
Ans:-

Manually Sending Data to a Page The last example for this chapter is a slight tangent to the other topics but plays off the idea of handling form data with PHP. As discussed in the section “Choosing a Form Method,” if a form uses the GET method, the resulting URL is something like http://www.example.com/page.php? some_var=some_value&age=20&... The receiving page (here, page.php) is sent a series of name=value pairs, each of which is separated by an ampersand (&). The whole sequence is preceded by a question mark (immediately after the

handling script's name). To access the values passed to the page in this way, turn to the $_GET variable. Just as you would when using $_POST, refer to the specific name as an index in $_GET. In that example, page.php receives a $_GET['some_var'] variable with a value of some_value, a $_GET['age'] variable with a value of 20, and so forth. You can pass data to a PHP script in this way by creating an HTML form that uses the GET method. But you can also use this same idea to send data to a PHP page without the use of the form. Normally you'd do so by creating links: Some Link That link, which could be dynamically generated by PHP, will pass the value 22 to page.php, accessible in $_GET['id']. To try this for yourself, the next pair of scripts will easily demonstrate this concept, using a hard-coded HTML page.

To create the HTML page: 1. Begin a new document in your text editor or IDE, to be named hello.html (Script 3.6):

Click a link to say hello:

2. Create links to a PHP script, passing values along in the URL:

- Michael
- Celia
- Jude
- Sophie

The premise here is that the user will see a list of links, each associated with a specific name A. When the user clicks a link, that name is passed to hello.php in the URL B. If you want to use different names, that's fine, but stick to one-word names without spaces or punctuation (or else they won't be passed to the PHP script properly, for reasons that will be explained in time). 3. Complete the HTML page:

To create the PHP script:

1. Begin a new document in your text editor or IDE, to be named **`hello.php`** (**Script 3.7**):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
  XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
  DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
  1999/xhtml" xml:lang="en"
 lang="en">
<head>
  <meta http-equiv="Content-Type"
  content="text/html;
  charset=utf-8"/>
  <title>Greetings!</title>
</head>
<body>
```

2. Begin the PHP code:

```
<?php // Script 3.7 - hello.php
```

3. Address the error management, if desired:

```
ini_set ('display_errors', 1); error_reporting (E_ALL | E_STRICT);
```

These two lines, which configure how PHP responds to errors, are explained in the pages leading up to this section. They may or may not be necessary for your situation but can be helpful.

4. Use the **name** value passed in the URL to create a greeting:

```
$name = $_GET['name']; print "<p>Hello, <span style=\"font-weight:
bold;\">$name</span>!</p>";
```

The **name** variable is sent to the page through the URL (see Script 3.6). To access that value, refer to **$_GET['name']**. Again, you would use **$_GET** (as opposed to **$_POST**) because the value is coming from a GET request.

Q 7)What is Validation? Explain Different types of validation in detail.

Ans:- In computer science, data validation is the process of ensuring that data have undergone data cleansing to ensure they have data quality, that is, that they are both correct and useful. It uses routines, often called "validation rules" "validation constraints" or "check routines", that check for correctness, meaningfulness, and security of data that are input to the system. The rules may be implemented through the automated facilities of a data dictionary,[1] or by the inclusion of explicit application program validation logic.
By Natasha Gilani

A validation check ascertains that the value (or data) input into a computer is valid. Validation checks are performed automatically by a computer to ensure that entered data is correct and reasonable. However, validation checks do not check whether or not the data entered is accurate. For instance, if a table accepts values ranging from 9 to 15, values greater or lower than this range are not accepted or validated by the computer, and an error is generated. There are numerous types of validation checks used to authenticate the correctness of entered data. They are used to authenticate online forms, email addresses and programming language syntax, and they are used by programs such as Microsoft Excel and Access.

Required Field Validation:-A required field validation check is commonly used in filling out online forms. It determines whether or not a user has filled out the important fields in a form. A required field validation check will not allow a form to be processed until all the mandatory fields are filled out. If one or more mandatory fields are left blank, a pop-up box appears, alerting the user to the missing fields.

Range Validation:-A range validation check determines whether or not an entered value falls within a predetermined range. For instance, if a text box allows values that fall within a set range -- say, from 5 to 10 -- all user-entered values that fall outside this range are not accepted. The range validation check alerts the user that the control value entered is inaccurate, so the user can make the necessary amendments. Range validation checks are also used to check the consistency of dates entered in forms.

Pattern Matching Validation:-Pattern matching validation is used to check whether or not data entered by a user follows a preset pattern. Email addresses are validated with pattern matching checks. Email validators check the syntax of an entered email address to determine whether or not it follows the SMTP protocol for email address standards. Email addresses are composed of two parts -- the local part and the domain name. For instance, abcd@example.com has "abcd" as its local part and "**example.com**" as the domain

name. Email pattern matching validation checks read and authenticate entered email addresses to check for syntax or other inconsistencies. **Abcd.example.com**, abc..1221@example.com, a@b@c@example.com are all invalid email addresses.

Numeric Validation:-Numeric validation checks determine whether or not data entered in a field or a text box has a numeric value (1, 2, 3, etc.), an alphabetic value (a, b, c, etc.) or an alphanumeric value (which includes symbols: &, %, #, etc.). Numeric validation checks allow users to only enter numeric values, and a pop-up alerts the user if he enters values other than numbers.

Unit 2

10 marks

1.Explain in detail PHP conditional Statement.

Ans:-The conditional statement is a key feature which allow us to perform the specific action on the basis of certain condition. Conditional statement are also known as the conditional expression or conditional constructs .Term conditional statements are usually use in imperative programing where condition change. The state of program, however , the term conditional expression or conditional functional programming.

The conditional statement in php are if ,elde, else if , and switch case .

1.If Statement:-We can use this control statement to execute some code only if a specified condition is true. The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluate, to FALSE.

2.If …..else Statement:-We use this control statement to execute some code only if a specified condition is true. elseif as its name suggest, is a combination o f if and else . Like else , it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE . However , unlike

3.if….else if….else Statement:- We use this control statement to select one of several books of code to be Executed .

4.Switch case:-We use this control statement to select one of many blocks of code to be executed.The switch case statement it is similar to IF statements on the same expression . In many occasion , you may want to compare the as,same variable , with many different values , and execute a different piece of coad depending on which value it equal to . This is exactly what the switch statement is for.

2.Explain different LOOPS in php with example.

Ans:-Loops are used to execute the same block of code a specified number of times.

1.While Loop:-PHP while loop is useful to execute a script until a specific condition remains true.

Example:-

```
<?php
$a=5;
$b=10;
While($a<$b)
{
Echo  “$a is less than $b ”.”<br/>”;
$a++;
}
?>
```

2.Do-while loop:-Do while loop first executes a piece of code once and then repeat the loop until the condition is satisfied . It executes statements once even if the condition is false.

Example:-

```
<?php
$a=5;
$b=10;
do
{
Echo  “$a is less than $b ”.”<br/>”;
$a++;
}
While ($a< $b
?>
```

3.For loop:-It execute a piece of code a specific number of times. PHP for loop execute a block of code of specified number of times. This loop is preferred when you are sure about. The number of iterations of the script.

Example:-

```
<?php
For($a=0;$a<=5;$a++)
Echo “the number is : $a<br>”;
}
?>
```

4.For each loop:-It is to execute a piece of code of each element in an array It is specially used for arrays and correspond to each key in an array.

Example:-

```
<?php
$cars=array (“”Skoda”, “BMW”;
Foreach  ($cars as $brand )

{
Echo “$brand <br>”;
}
?>
```

3.What is array ? Explain diffrent type of array in detail with example.

Ans:-Php array is an ordered map content value on the bases of a key. It is use to hold multiple value of similar type in asingle variable.

There are three types of array in php:-

1.Index array:-Index is represented by index number which starts from 0.We can store number string an object in the php array. All php array elements are assign to an index number by default.

Example:-

```
<?php
$season=array(“Summer”
                “Winter”
                “Rainy”);
Echo “seasone  are : $seasone[0] , $seasone [1] , $seasone [2]”;
?>
```

2.Associative array:-We can associate name with each array elements in php using (□).In the product array,We allowed php to give each item the default index. This meant the first item,we added became 0,The second item 1, and so on.php also supports associative array,In an associative array,we can associate any key or index we want with each value.

Example:-<?php

```
$salary=array(“abc”□”5000”,
                ”pqr”□”15000”,
                “xyz”□”10,000”);
  Echo $salary are :-;
  $salary[abc],$salary[pqr],$salary[xyz];
   ?>
```

3.Multidimansional array:-php multidimensional array is also known as array of array .if allow you to store tabular data in an array. Php multidimensional array can be represented in the form of matrix which is represented by row*column.

Example:-

In the product array,We allowed php to give each item the default index. This meant the first item,we added became 0,The second item 1, and so on.php also supports associative array,In an associative array,we can associate any key or index we want with each value.

Example:-

```
<?php
$temp=array
(
array(“Sonu”,4.00.000);
array (“Jhon”5.00.0000);
array( “Kartik”1.00.000);
)
For($row=0;$row<3;$row++)
{
For($col=0;$col<3;$col++)
{
Echo $temp[$row] [$col];
}
Echo “<br/>”;
}
?>
```

4.Explain sorting array in detail.

Ans:-php has several functions that deal with sorting array, and this document exist to help sort it all out.PHPprovides a range of functions for sorting database on either the key or value of an associative array. Where its gets complicated is when you need to sort an array of associative array by one one or more conditions.This is a very common task for php programmer as data returns from database /SQL queries often appears in associative array format. Some sort base on array keys, whereas other by the values. Whether or not the correlation Between the keys and values are maintained after sort , which may mean the keys are reset numerically.

5.Explain any five string function in php.

Ans1.**php**strtolower():-the strtolower()function returns string to lowercase letters.

This function binary safe.

Example:-<php
$str =”My name is Khan”;
$str =”strtolower($str);
Echo $str;
?>

2.phpstrtoupper():- The string strtoupper() function return string in uppercase to letter .This function is binary-safe. Returns string with all alphabetic characters converted to uppercase.

Example:- :- <php
$str =”My name is Khan”;
$str =strtoupper($str);
Echo $str;
?>

3.phpUCfirst():-The uc function return string converting first character into appear. It does change the can of other change.

Example:- :- <php
$str =”My name is Khan”;
$str =UCfirst($str);
Echo $str;
?>

4.phplcfirst():-The lcfirst() function return string character into lowercase. It doesn’t change the case of other character.

Example:- :- <php
$str =”My name is Khan”;
$str =lcfirst($str);
Echo $str;
?>

5.Ucwords () function:-The ucwords () function returns string converting first character of each word into uppercase.

Example:- :- :- <php
$str =”My name is Khan”;
$str =ucwords($str);
Echo $str;
?>

6.What is function in php ? Explain different types of function with example.

Ans:-Phpuser define functions, Besides the built - in php function .We can create our own functions. A functions is a block of statements that can be used repeatedly in a program . A function will not execute immediately when a page loads. A functions will be executed by a call to the function.

Different Types of function:-

1.PHP Function with Parameters:-PHP gives you option to pass your parameters inside afunction . you can pass as many as parameters your like. These parameters work like variables inside your function .

Example:-

<?php
Function addFunction ($num1, $num2)
{
$sum=$num1 + $num2;

```
Echo “Sum of the two numbers is: $sum”;
}
addFunction(10, 20);
?>
```

2.Parametrised Function:-Parameterized function are the functions with parameters you can pass any numbers of parameters inside a function. This pass parameter act as variable inside your function. They are specified inside the parenthesis after the function name. the output depend upon the dynamic value pass as the parameter into the functions.

Example:-

```
<?php
Function add1($x,$y)
{
Echo “addition of two number is:”$add1;
$add1 $x+$y;
}
Add1(10 , 20);
?>
```

3.Recursive Function:- Php supports recursive function call like c or c++ . In such case we call current fuction within function it call as recursive function. It is recommended to avoid recursive function call over +200 recursive leval because it may smash the stage and may cause the termination of script.

Example:-

```
<?php
Function add1($num)
{
If($num<=5)
{
Echo $num<br/>;
Add ($num +1);
}
Add1(1);
?>
```

4.Returning value:-Values are returned by using the optional return statement . Any types may be returned, includes arrays and object . This causes the function to end its execution immediately and pass control back to the line from which it was called .

Example:-

```
<?php
Function cube ($x)
{
Return $x*$x*$x*;
}
Echo “Cube of is ” cube (5);
?>
```

7.Explain If…..else statement in php with example.

Ans:-Ans:- php conditional statment .Very often when you write code, you want to perform different action for diffrentconditions.you can use conditional statments your code to do this if ....else statment execute some code if a condition is true and another code if condition is false.

Example:-

```
<?php
$num=100;
If($num%2=0)
{
Echo “$num is even number”;
}
Else
{
Echo ”$num is not even number”;
}
?>
```

8.Explain switch statement with example.

Ans:- :-The switch statement is used to perform different action based on different conditions.Use the many blocks of code to be executed .The switch statement is similar to if statements on the same expression.In many occasions, you may want to compare the same variable with many different values, and execute a different piece of code depending on which value it equals to.

Example:-

```
<?php
$num=20;
Switch ($num)
{
Case 10;
Echo (“number is equals to 10”);
Braek ;
Case 20:
Echo (“number is equal to 30”);
Break;
Default;
Echo (“number is not equal to 10,20,30”);
}
?>
```

9.Explain index array and associative array in detail.

Ans:- 1.Index array:-Index is represented by index number which starts from 0.We can store number string an object in the php array. All php array elements are assign to an index number by default.An index or numeric array store each array element with numeric index.

2.Associative array:-We can associate name with each array elements in php using ().In the product array,We allowed php to give each item the default index. This meant the first item,we added became 0,The second item 1, and so on.php also supports associative array,In an associative array,we can associate any key or index we want with each value.

10.Explain for each loop in detail.

Ans:-For each construct provide an easy way toitrerate over arrays. For each works only on arrays and objects, and will issue an error when you try to use it on a variable in a different data type or an uninitialized variable .The first form loop over the array given by array expression .on each iteration , the value of a current elements in assigned to value of the current elements in assigned to $value and the

internal array pointer is advanced by one.In order to be able to directly modify array elements within the loop precede $value with &.In that case the value will be assigned by reference.

Part - C (Each question carries Ten marks)

UNIT – III

1. Explain date function with 10 formatting characters.

Answer:

date() Function

Example

Format a local date and time and return the formatted date strings:

```
<?php
// Prints the day
echo date("l") . "<br>";

// Prints the day, date, month, year, time, AM or PM
echo date("l jS \of F Y h:i:s A");
?>
```

Definition and Usage

The date() function formats a local date and time, and returns the formatted date string.

Syntax

date(*format,timestamp);*

Parameter	Description
Format	Required. Specifies the format of the outputted date string. The following

characters can be used:

- d - The day of the month (from 01 to 31)
- D - A textual representation of a day (three letters)
- j - The day of the month without leading zeros (1 to 31)
- l (lowercase 'L') - A full textual representation of a day
- N - The ISO-8601 numeric representation of a day (1 for Monday, 7 for Sunday)
- S - The English ordinal suffix for the day of the month (2 characters st, nd, rd or th. Works well with j)
- w - A numeric representation of the day (0 for Sunday, 6 for Saturday)
- z - The day of the year (from 0 through 365)
- W - The ISO-8601 week number of year (weeks starting on Monday)
- F - A full textual representation of a month (January through December)
- m - A numeric representation of a month (from 01 to 12)
- M - A short textual representation of a month (three letters)
- n - A numeric representation of a month, without leading zeros (1 to 12)
- t - The number of days in the given month
- L - Whether it's a leap year (1 if it is a leap year, 0 otherwise)
- o - The ISO-8601 year number
- Y - A four digit representation of a year
- y - A two digit representation of a year
- a - Lowercase am or pm
- A - Uppercase AM or PM
- B - Swatch Internet time (000 to 999)
- g - 12-hour format of an hour (1 to 12)
- G - 24-hour format of an hour (0 to 23)
- h - 12-hour format of an hour (01 to 12)
- H - 24-hour format of an hour (00 to 23)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds, with leading zeros (00 to 59)
- u - Microseconds (added in PHP 5.2.2)
- e - The timezone identifier (Examples: UTC, GMT, Atlantic/Azores)
- I (capital i) - Whether the date is in daylights savings time (1 if Daylight Savings Time, 0 otherwise)
- O - Difference to Greenwich time (GMT) in hours (Example: +0100)

- P - Difference to Greenwich time (GMT) in hours:minutes (added in PHP 5.1.3)
- T - Timezone abbreviations (Examples: EST, MDT)
- Z - Timezone offset in seconds. The offset for timezones west of UTC is negative (-43200 to 50400)
- c - The ISO-8601 date (e.g. 2013-05-05T16:34:42+00:00)
- r - The RFC 2822 formatted date (e.g. Fri, 12 Apr 2013 12:01:05 +0200)
- U - The seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)

and the following predefined constants can also be used (available since PHP 5.1.0):

- DATE_ATOM - Atom (example: 2013-04-12T15:52:01+00:00)
- DATE_COOKIE - HTTP Cookies (example: Friday, 12-Apr-13 15:52:01 UTC)
- DATE_ISO8601 - ISO-8601 (example: 2013-04-12T15:52:01+0000)
- DATE_RFC822 - RFC 822 (example: Fri, 12 Apr 13 15:52:01 +0000)
- DATE_RFC850 - RFC 850 (example: Friday, 12-Apr-13 15:52:01 UTC)
- DATE_RFC1036 - RFC 1036 (example: Fri, 12 Apr 13 15:52:01 +0000)
- DATE_RFC1123 - RFC 1123 (example: Fri, 12 Apr 2013 15:52:01 +0000)
- DATE_RFC2822 - RFC 2822 (Fri, 12 Apr 2013 15:52:01 +0000)
- DATE_RFC3339 - Same as DATE_ATOM (since PHP 5.1.3)
- DATE_RSS - RSS (Fri, 12 Aug 2013 15:52:01 +0000)
- DATE_W3C - World Wide Web Consortium (example: 2013-04-12T15:52:01+00:00)

Timestamp Optional. Specifies an integer Unix timestamp. Default is the current local time (time())

Technical Details

Return Value: Returns a formatted date string on success. FALSE on failure + an E_WARNING

PHP Version: 4+

Changelog: PHP 5.1.0: Added E_STRICT and E_NOTICE time zone errors. Valid range of timestamp is now from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT. Before version 5.1.0 timestamp was limited from 01-01-1970 to 19-01-2038 on some systems (e.g. Windows).
PHP 5.1.1: Added constants of standard date/time formats that can be used

to specify the format parameter

2. Explain in detail cookies with example.

Answer:

A cookie is often used to identify a user.

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
```

```
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

3. Explain in detail session with example .

answer:

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.

What is a PHP Session?

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.

So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a database.

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

How does it work? How does it know it's me?

Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans

the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>
```

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

4. What is function? Explain types of function in detail.

answer:

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

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

Create a User Defined Function in PHP

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

Syntax

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

Note: A function name can start with a letter or underscore (not a number).

Tip: Give the function a name that reflects what the function does!

Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

Example

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```
<?php
function familyName($fname) {
  echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

The following example has a function with two arguments ($fname and $year):

Example

```
<?php
function familyName($fname, $year) {
  echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<?php
function setHeight($minheight = 50) {
  echo "The height is : $minheight <br>";
}
```

```
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions - Returning values

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

Example

```
<?php
function sum($x, $y) {
   $z = $x + $y;
   return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

ouput:

5 + 10 = 15
7 + 13 = 20
2 + 4 = 6

5. How to create, modify, delete cookies in PHP?

answer:

A cookie is often used to identify a user.

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
   echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
   echo "Cookie '" . $cookie_name . "' is set!<br>";
   echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

6. How to create ,modify, delete session in PHP? Explain.

answer: A cookie is often used to identify a user.

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
```

```
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

7. Explain in detail validation functions in PHP.

Answer:

This and the next chapters show how to use PHP to validate form data.

PHP Form Validation

Think SECURITY when processing PHP forms!

These pages will show how to process PHP forms with security in mind. Proper validation of form data is important to protect your form from hackers and spammers!

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

PHP Form Validation Example

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male *

Your Input:

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Big Note on PHP Form Security

The $_SERVER["PHP_SELF"] variable can be used by hackers!

If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

So far, so good.

However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.

Be aware of that any JavaScript code can be added inside the <script> tag! A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

How To Avoid $_SERVER["PHP_SELF"] Exploits?

$_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post"
action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

The exploit attempt fails, and no harm is done!

Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example

```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Notice that at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

However, in the example above, all input fields are optional. The script works fine even if the user does not enter any data.

The next step is to make input fields required and create error messages if needed.

8. Explain include and require functions in detail.

Answer:

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax

include '*filename*';

or

require '*filename*';

PHP include Examples

Example 1

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

To include the footer file in a page, use the include statement:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

Example 2

Assume we have a standard menu file called "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

All pages in the Web site should use this menu file. Here is how it can be done (we are using a element so that the menu easily can be styled with CSS later):

Example

```
<html>
<body>

<div class="menu">
<?php include 'menu.php';?>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>

</body>
</html>
```

Example 3

Assume we have a file called "vars.php", with some variables defined:

```
<?php
$color='red';
$car='BMW';
?>
```

Then, if we include the "vars.php" file, the variables can be used in the calling file:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'vars.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

PHP include vs. require

The require statement is also used to include a file into the PHP code.

However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
```

```
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

Use require when the file is required by the application.

Use include when the file is not required and application should continue when file is not found.

9. What is call by value and call by reference? Explain with example.

Answer:

PHP Call By Value

PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.

Let's understand the concept of call by value by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Value' string. But, printing $str variable results 'Hello' only. It is because changes are done in the local variable $str2 only. It doesn't reflect to $str variable.

```
1. <?php
2. function adder($str2)
3. {
4.     $str2 .= 'Call By Value';
5. }
6. $str = 'Hello ';
7. adder($str);
8. echo $str;
9. ?>
```

Output:

```
Hello
```

Example 2

Let's understand PHP call by value concept through another example.

```
1. <?php
2. function increment($i)
3. {
4.     $i++;
5. }
6. $i = 10;
7. increment($i);
8. echo $i;
9. ?>
```

Output:

```
10
```

PHP Call By Reference

In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.

Let's understand the concept of call by reference by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Reference' string. Here, printing $str variable results 'This is Call By Reference'. It is because changes are done in the actual variable $str.

```
1. <?php
2. function adder(&$str2)
3. {
4.     $str2 .= 'Call By Reference';
5. }
6. $str = 'This is ';
7. adder($str);
8. echo $str;
9. ?>
```

Output:

```
This is Call By Reference
```

Example 2

Let's understand PHP call by reference concept through another example.

```
1. <?php
2. function increment(&$i)
```

```
3. {
4.     $i++;
5. }
6. $i = 10;
7. increment($i);
8. echo $i;
9. ?>
```

Output:

```
11
```

10. What is filter_var() function ? Write a program to validate email-id with proper message.

Answer:

Definition and Usage

The filter_var() function filters a variable with the specified filter.

Syntax

filter_var(*var, filtername, options*)

Parameter	Description
var	Required. The variable to filter
filtername	Optional. Specifies the ID or name of the filter to use. Default is FILTER_DEFAULT, which results in no filtering
options	Optional. Specifies one or more flags/options to use. Check each filter for possible options and flags

Technical Details

Return Value: Returns the filtered data on success, or FALSE on failure
PHP Version: 5.2.0+

PHP filter_var() Function

Example

Check if the variable $email is a valid email address:

```
<?php
$email = "john.doe@example.com";
```

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
  echo("$email is a valid email address");
} else {
  echo("$email is not a valid email address");
}
?>
```

output:

john.doe@example.com is a valid email address

PHP 5 Forms - Validate E-mail and URL

This chapter shows how to validate names, e-mails, and URLs.

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters and whitespace. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/
%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

PHP - Validate Name, E-mail, and URL

Now, the script looks like this:

Example

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

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

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression also allows dashes in the
URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/
%=~_|]/i",$website)) {
```

```
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

output:

PHP Form Validation Example

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male *

Your Input:

Unit 4
10 marks

1.Explain MySQL privileges in detail.

Ans:-The privileges granted to a MySQL determine which operation the account can performs . Database privileges apply to a database and to all objects within it . This privileges can be granted for specific database , or globally so that they apply to all database . Administrative privileges enable users to manage operation of the MySQL server . These privileges are global because they are not specific to a particular database. Database privileges apply to a database and to all object within it .This privileges can be granted for specific database. And globally so that they apply to all databases .privileges for database object such as table , index ,views , and stored runtimes can be granted for all objects of a given type within a

database .Privileges Supported by MySQL .The following table summarizes the permissible priv_type privilege types that can be specified for the grant and REVOKE statements , and the levels at which each privilege , can be granted . for additional information about each privilege , see section 6.2.1,Privileges provided by MySQL.

2.Write a note on MySQL error Handling.

Ans:-Improved your store procedure Error Handling with GET DIAGNOSTICS .By Chris Calender. In a previous post , I discussed debugging stored procedures with RESIGNAL , which is of great values when troubleshooting errors raised by your stored procedures , functions ,triggers, and events as of MySQL .If you get error in stored procedure, instead of an exist , you should continue without any error message that means you can show any default or custom error code or message to the application/user.

3.Explain any five date functions in MySQL.

Ans:-:- 1.CURRENT_DATE():-in MySQL , the CURRENT_DATE returns the current date in 'YYYY-MM-DD' format or YYYYMMDD format depending on whether numeric and string is used in function .

CURDATE() and CURRENT _DATE() are the synonym of CURRENT_DATE.

2.DATE_ADD ():- This functions perform date arithmetic . date is a DATETIME or DATE value specifying the string date . expression specify the interval value to be added or subtracted from the starting date.

3.DATE_FORMAT():-formats the date values according to the format string .The following specifies may be used in the format string. The % characters required before format specified characters.

4.CURDATE() :-Returns the current date as a value in 'YYYY-MM-DD' format , depending on whether the function is used in a string or numeric context.

5.DATEDIFF(expr1 , expr 2):-DATEDIFF () returns expr1, expr2 expressed as a values in days from one date to other . expr1, and expr2 are date or date-and-time expressions.

4.What is pattern Matching in MySQL ? Explain.

Ans:- MySQL provides standard SQL pattern matching as well as a form of pattern matching based on extended regular expression similar to those use by unix utilities such as.SQL patter matching enable you to use match any single character and % to match an arbitrary number of characters . ImMySQL ,SQL pattern are case-insensitive by default. MySQL provide standard SQL pattern matching as well as form of pattern matching based on extended regular expressions similar to those used by unix utilities such as vi,grep and sed.SQL patter matching enables you to use to match any single character and % to match an aribitary number of characters .

5.How to connect php with MySQL ? Expalin.

Ans:-To connect to MySQL using PDO , follow these Steps:-

Use the following php code to connect to MySQL and select a database . Replace username with with your username , password with your password , and dbname with the database name:

```
<?php $my PDO =new
PDO ('mysql:host=localhost ;dbname =dbname','username','password');
?>
```

6.Explain the process of creating Table , Delating table in mysql.

Ans:-Create Table process:-

1.Open your database .In order to create a table, you must have a database to house it in…..
2.Learn the basic data types. Every entry in the table is store as a certain type of data….
3.Create you table..
4. Verify that your table was created properly.
5.Create a table using php.
Delete Table process:-
"To delete a table , type the following command from the mysql> prompt . replace table name with the name of the table that you want to delete: DROP TABLE tablename: this command assumnes that you have already selected a database by using a USE statement".

7.Explain any ten data types in MySQL with example .
Ans:-1.IntA normal-sized integer that can be signed or unsigned .if signed , the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
Example:-
2.TINYINT:-A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.

- **SMALLINT** − A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** − A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** − A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** − A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will

default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

- **DECIMAL(M,D)** − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.