SPPU New Syllabus

A Book Of

WEB TECHNOLOGY

For BCA (Science) : Semester - IV

[Course Code BCA242]

CBCS Pattern

As Per New Syllabus, Effective from June 2020

Mr. Abhijeet D. Mankar
M.C.S, SET,
Assistant Professor,
Department of Computer Science,
Tuljaram Chaturchand College of Arts,
Science and Commerce (Autonomous),
Baramati.

Mrs. Swati S. Jadhav
M.C.S, NET, M.Phil,
Assistant Professor,
Department of Computet Science,
MES Abasheb Garware College

Mr. Gajanan A. Deshmukh
B.Sc (Comp.sci), MCA
HOD: BCA DEPT.
Smt. Kashibai Navale College of Commerce, Pune -04.

Price ₹ 250.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N5552

Syllabus ...

1. **Introduction to PHP** **[6 Hrs]**
   - HTTP basics, Introduction to web server and web browser
   - Introduction to PHP
   - PHP - Lexical structure, Language basics.
2. **Functions and Arrays** **[8 Hrs]**
   - Defining and calling a function
   - Default parameters
   - Variable parameters, Missing parameters
   - Variable function, Anonymous function
   - Indexed Vs Associative arrays
   - Identifying elements of an array
   - Storing data in arrays
   - Multidimensional arrays
   - Extracting multiple values
   - Traversing arrays
   - Sorting Using arrays
3. **Object Oriented Programming** **[10 Hrs]**
   - Classes
   - Objects
   - Introspection
   - Serialization
   - Inheritance
   - Interfaces
   - Encapsulation
4. **Web Techniques** **[12 Hrs]**
   - Variables
   - Server information
   - Processing forms
   - Setting response headers
   - Maintaining state
   - SSL

5. Databases **[12 Hrs]**

- Using PHP to access a database
- Relational databases and SQL
- PEAR DB basics
- Advanced database techniques

6. Ajax and XML **[12 Hrs]**

- Understanding java scripts for AJAX
- AJAX web application model
- AJAX–PHP framework
- Performing AJAX validation
- Handling XML data using php and AJAX
- What is XML?
- XML document Structure
- PHP and XML
- XML parser
- The document object model
- The simple XML extension
- Changing a value with simple XML

***

Contents ...

1. Introduction to PHP 1.1 - 1.36

2. Functions and Arrays 2.1 - 2.48

3. Object Oriented Programming 3.1 - 3.26

4 Web Techniques 4.1 - 4.30

5. Databases 5.1 - 5.44

6. Ajax and XML 6.1 - 6.52

***

1... Introduction to PHP

Objectives...

- To know about the basics of HTTP.
- To get familiar with the features of PHP.
- To have knowledge about the PHP language basics.
- To introduce concept of web server and web browser.

1.1 INTRODUCTION

- Tim Berners-Lee invented the term World Wide Web in 1989 which is popularly known as WWW. At the time of writing this, as per the latest information on the WWW, PHP is present in the raking of top 10 most used programming languages by developers worldwide.
- Searching on Google using "PHP Programming" results in huge amount of pages. It shows the popularity of this scripting language. The main reason behind its popularity: "PHP is fast and easy for use and development". The server stores the web pages.
- A website is a collection of web pages and other related content that is identified by a domain name.

1.2 HTTP BASICS

- HTTP (Hypertext Transfer Protocol) is one of the most popular application protocol used in the internet. Client sends its request and in return receives response from the server.
- The interaction between a client and server is somewhat similar to the interaction between a customer and a waiter at a restaurant. Here, a waiter acts like a server in the restaurant.

- In the best case, waiter may find the menu item available in the kitchen which was ordered by the customer. The waiter then serves the menu item to the customer. It is similar to the case where the server finds the data/resource requested by the client and responds the client with the appropriate data/resource.
- In the worst case, waiter may not find the menu item ordered by the customer in the kitchen. The waiter then politely informs the customer about unavailability of the menu item. It is similar to the case where the server does not find the data/resource requested by the client and responds the client with the appropriate message indicating that the resource requested is not found. The interaction between client and server is depicted by using following figure.

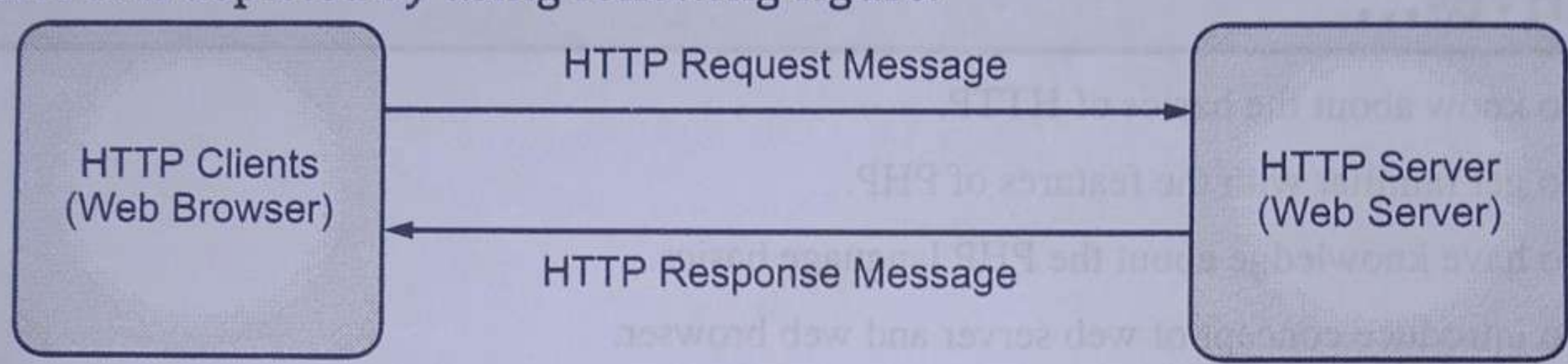


Fig. 1.1 Interaction between HTTP Clients and HTTP Server

Features of HTTP (Hypertext Transfer Protocol):

- HTTP is one of the popular application layer protocols.
- HTTP is connectionless. The client and server know each other during the current request and response only, after which the client disconnect the connection without waiting for any new requests from the same client to the same server.
- HTTP is stateless. The client and the server know each other during the current request only, after which they disconnect. Because of this, neither client nor the browser can preserve information between different requests across the web pages. It simply means that the current request does not know what has been done in the previous requests.

1.2.1 HTTP Request and Response

- **Format of HTTP request message is:**
  - Request Message Header
    - Request Line
    - Request Headers
  - Blank Line
  - Request Message Body (optional)
- **Syntax of Request Line :** `method_name request_URI HTTP version`

  For example,

```
GET/abc.html HTTP/1.1
POST/hello.html HTTP/1.1
```

- **Syntax for Request header:** `header_name : value1, value3, …`

  For example,

  ```
  Host : www.abc.org
  Connection : Keep-Alive
  ```

- **Request Message Body:** In case of GET method, the HTTP request body is empty. With the POST method, it contains additional data.
- **Format of HTTP response message is:**
  - Response Message Header
    - Status Line
    - Response Headers
  - Blank Line
  - Response Message Body (optional)
- **Syntax for Status Line:** `HTTP-version response-status-code reason-phrase`

  Where,

  `response-status-code`: Three digit number generated by the server to reflect the outcome of the request.

  `reason-phrase`: Short explanation to the status code.

  For example,

  ```
  HTTP/1.0 200 OK
  HTTP/1.0 404 Not Found
  ```

- **Syntax for Response Header:** `Header_name : value1, value2, …`

  For example,

  ```
  Content-Type : text/html
  Connection : Keep-Alive
  ```

- **Response Message Body:** If the response was successful, the HTTP response body contains the HTML code which is ready for the browser interpretation. A failure code is sent if the response is unsuccessful.

1.2.2 Introduction to Web Server and Web Browser

- **Server:** A server is responsible for accepting a request from client and sending response back to the client through a computer network. There are different types of servers.
  - **Proxy Server:** It works in between client and an external server to filter requests.
  - **Mail Server:** It is used to move and store mail over corporate networks, internet.
  - **Web Server:** It delivers the web content to the browser.
  - **Application Server:** It connects database servers and end users (middleware).

- **Telnet Server:** It enables users to log on to a host computer.
- **List Server:** It manages mailing lists (newsletters or advertising).
- **FTP Server:** It moves one or more files securely between computers.
- **Virtual Server:** It provides cheap web hosting services (dozens of virtual servers can co-exist on the same computer).
- **IRC Server:** It is an option for real-time discussion capabilities. Internet Relay Chat consist different network servers that enable the users to connect with each other through an IRC network.

- **Web Server:** Web Server helps to deliver the web content which can be accessed through internet. Popular web servers are listed below.
  - **Apache HTTP Server:** Apache Software Foundation

    The Apache HTTP Server Project provides a secure, efficient and extensible server that provides HTTP services synchronous with the latest HTTP standards. The Apache HTTP Server was launched in 1995. It is known as httpd.
  - **Nginx("engine-x"):** NGINX, Inc.

    Nginx("engine-x") was released in 2004 by Igor Sysoev. After its release, it was used as a load-balancer or reverse proxy in front of Apache installations.
  - **Lighttpd (pronounced "lighty"):** Open Source

    It is designed and optimized for higher performance environments. Lighttpd is the solution for the servers suffering from load problems.
  - **GWS (Google Web Server):** Google

    Google Web Server is a proprietary web server software which Google uses for its own web infrastructure. It is exclusively used inside Google's ecosystem.
  - **Internet Information Services (IIS):** Microsoft

    Internet Information Services (IIS) is a web server that runs on the Microsoft .NET platform on the Windows OS. It is a flexible, manageable and safe web server for hosting anything on the web.
- **Web Browser:** Web Browser is an application which helps us to retrieve, present, traverse information resources on the WWW. Some examples of Web browsers are Google Chrome, Mozilla Firefox, Microsoft Edge, Opera, Safari etc. Web Browser has a user interface. Let us take a look on how browser works:

  1. We type a URL in the browser's URL address bar.
  2. The browser then communicates with the web server to find the page requested by the client. It also requests the page's information to the Web server.
  3. The browser then receives a file in the form of HTML or JavaScript code containing instructions regarding how to render the information found on that page.
  4. The browser then interprets the file and renders (displays) the page for the client.

1.3 INTRODUCTION TO PHP

- The official full form of the PHP is **"Hypertext Preprocessor"**. In the beginning years of PHP it was known as "Personal Home Page" and is still known by this name. It is one of the popular server side scripting languages.

1.3.1 History of PHP

- The history of PHP starts from the year 1994. We summaries some of the versions in the following table.

Table 1.1: Versions of PHP

PHP Version	Month and Year of Release	Description
PHP 1.0	June 1995	Rasmus Lerdorf made announcement about PHP 1.0. Officially it was called "Personal Home Page Tools(PHP Tools)"
PHP Version 2(PHP/FI) "Personal Home Page/Forms Interpreter" announcement	November 1997	Officially named "PHP/FI 2.0".
PHP 3.0	June 1998	Zeev Suraski and Andi Gutmans collaborated with Rasmus Lerdorf to rewrite the underlying parsing engine.
PHP 4.0.0	May 2000	Advance parser written by Zeev and Andi, which was known as Zend Engine 1.0.
PHP 5.0.0	July 2004	New Zend Engine II introduced.
PHP 7.0.0	December 2015	Addition of ?? (null coalescing) and <=> (spaceship) operators etc.
PHP 7.1.0	December 2016	Addition of void return type etc.
PHP 8.0	November 2020	Addition of new features including named arguments, union types etc.

1.3.2 What does PHP Do?

- The three main areas where PHP is used are as given below:
  1. **Server Side Scripting:** This is the main target field for PHP. PHP is designed to create dynamic web content. Three things are needed to use PHP for server side scripting. These are PHP parser, Web Server and Web Browser. PHP has been popular for generating XML documents, graphics, Flash animations, PDF files and much more.
  2. **Command-Line Scripting:** PHP can also be used for command-line scripting like Perl, Awk or the UNIX shell. It can be used for system administration tasks like backup.
  3. **Writing Desktop Applications (Client Side GUI Applications):** PHP-GTK can be used to write cross platform GUI applications.

1.3.3 Why PHP is better?

- It is interesting to know the reasons why people choose PHP. Some of the reasons are listed below:
  1. It is easier to code and also faster to execute the code in PHP.
  2. PHP is platform independent. It means that the code written in PHP can run on any operating System (UNIX, Linux, Windows, Mac OS X etc.).
  3. PHP is free. One can visit the URL `http://www.php.net` for download.
  4. Besides Apache Web Server, PHP is compatible with other web servers like Microsoft Internet Information Server.
  5. PHP is capable for adapting future extension of functionality.
  6. PHP supports many databases (backends) such as PostgreSQL, MySQL, Oracle, Informix, Sybase, MongoDB etc.

1.3.4 How PHP works with the Web Server?

- A web browser makes a request for a Web page. Depending on the URL, the browser finds the address of Web server. It then identifies the page and gives any other information the Web server may require. The other information given to the Web server may contain the text typed by the user into the form fields.
- In case of HTML file request, the Web server identifies the file, informs the browser to expect some HTML text and finally send the contents. Upon receiving the contents, the browser starts rendering the page.
- When a Web server receives a request for a CGI, it cannot send just the contents of file. It first executes the script, which generates some HTML code, which is further sent to the browser. From the point of view of browser, it is getting HTML. In case of a PHP page request, the process remains the same like a CGI, at least to the extent that the script is not sent to the browser. It first passes through the PHP engine, which gives the web server HTML text.

1.4 PHP - LEXICAL STRUCTURE

- The lexical structure is the set of basic rules that regulate how you design programs in that specific language.
- There are various aspects related with lexical structure.

1. **Case Sensitivity:** In PHP, some terms are case sensitive while some are not.

Table 1.2: Case Sensitive / Insensitive PHP Terms

Case Sensitive	Case Insensitive
Variables are case sensitive.	Names of user-defined classes, functions and keywords like echo, while etc. are case insensitive.
For example, $name, $NAME, $NaME are treated as three different variables.	For example, echo "hello, world"; ECHO "hello, world"; EcHo "hello, world"; The above three statements are equivalent.

2. **Statements and Semicolons:** A statement is a collection of PHP code that does some operation. A statement can be an assignment statement in the simple case or a loop with different exit points in a complex case.
3. **Whitespace and Line Breaks:** Whitespace does not play any role in a PHP program. For example,

```
calculate_premium ($amount, $tenure, $interest_rate);
```

Could be written as follows,

```
calculate_premium ($amount,
$tenure,
$interest_rate);
```

4. **Comments:** Comments can be given in PHP using three different ways.

Table 1.3: Comment Types

Comment style	Description	Example
Multiline Comments	It is used to give multiline comments. **Syntax**: /* comment */	/* This function calculates factorial of a number */ $f = fact($num);
Single Line Comments	It is used to give single line comment. **Syntax** : // comment	// factorial of a number $f = fact($num);
	It is same as C++ style comments and often used in Unix shell scripting. **Syntax** : # comment	# factorial of a number $f = fact($num);

5. **Identifiers:**
   - Identifiers are used to name variables, constants, functions and classes.
   - For Example,

     `$amount, $_cnt, $grand_total` are all valid variable names.

     `$in valid, $3a` are invalid variable names.
   - A **Constant** is an identifier for a value. More importantly, once it is set, we cannot change its value. In PHP, we use define function to define a constant.

     For Example,

```
define('publ',"Nirali Prakashan");
echo publ;
```

6. **Literals:** A data value which appears directly in a program is called as a Literal. For example: `2007, 1.34, "Good Morning", 'Hello', true` are all literals.
7. **Keywords/ Reserved words:** A keyword/reserved word is a word whose meaning is reserved by the language for its functionality. Keywords in PHP are case insensitive.

1.5 LANGUAGE BASICS

1.5.1 Data Types

- There is no explicit data type declaration for variables in PHP. The value assigned to a variable decides its data type. In PHP, the data types are broadly categorized into three main categories, namely:
  1. **Single value:** Integers, Floating point numbers, Strings, Booleans.
  2. **Collection (compound):** Arrays, Objects.
  3. **Special:** Resource, Null.

Single value Data types:

1. **Integers:**

- The whole numbers such as 1, 11 are integers. The range of integer is platform dependent. On 32 bit platform the range of integers is from –2,147,483,647 to +2,147,483,647.
- Integer size can be found using constant PHP_INT_SIZE. To find maximum value constant PHP_INT_MAX can be used. Specifying a number beyond the acceptable range results in interpreting it as a floating point number.
- Integers can be written in decimal, hexadecimal, octal or binary format.
  - Example of **decimal integers** are 1978, -21, +52. By default positive sign is considered.
  - **Hexadecimal numbers** begin with 0x. It is followed by 0-9 (digits) or A-F (letters can be in uppercase or lowercase). Examples of hexadecimal numbers are as follows:

    0xAF // decimal equivalent 175

    0xC //decimal equivalent 12

- **Octal numbers** precede(start) with a zero. Examples of octal numbers are as follows:

  025 //decimal equivalent 21

  0171 //decimal equivalent 121

- **Binary numbers** precede (start) with 0b followed by sequence of bits (consisting of 0 and 1).

  0b10 //decimal equivalent 2

- In PHP, there is no data type declaration for a variable. In order to inquire about whether the variable contains an integer value or not we use Is_Int( ) (or is_integer( ))function.

Program 1.1: Write a program in PHP which checks whether a given value is an integer or not.

```
<?php
$a=10;
if(is_int($a))
   echo "$a is an integer";
else
   echo "$a is not an integer";
?>
```

Output: 10 is an integer.

2. Floating point numbers:

- The numeric values with decimal digits are floating point numbers. This allows numbers between 1.7E-308 and 1.7E+308 with 15 digits of accuracy.
- There are two formats of floating point numbers:

  (i) Regular format : 0.11, 3.14, 0.05

  (ii) Scientific format : 11.0E-3 (0.011 in Regular format).

- The `is_float()` (or `is_real()`) function is used to check whether a value is floating point number.

3. Strings :

- A sequence of characters of any random length is known as a String.
- There are four ways to specify strings in PHP : Single quoted strings, Double quoted strings, Nowdoc, Heredoc.
- For Example, `'apple'`, `"apple"`. The value of a variable is replaced if it is enclosed within double quotes, but this does not happen in case of single quotes. This is known as variable interpolation.

Program 1.2: Write a PHP program which demonstrates variable interpolation.

```
<?php
$fruit="apple";
echo "$fruit<br>";
echo '$fruit';
?>
```

Output:

apple

$fruit

- Double quotes support a number of escape sequences as compared to single quotes. The escape sequences supported by double quoted strings are as follows:

\n : Newline	\" : double quotes
\t : Tab	\r : Carriage return
\\ : Backslash	\$: dollar sign
\{ : Left Brace	\} : Right Brace
\[: Left Bracket	\] : Right Bracket

\0 to \777 : ASCII character represented by octal value

\x0 to \xFF : ASCII character represented by hexadecimal value

A single quoted string allows \\ to get a literal backslash and \' to get a literal single quote

Program 1.3: Write a PHP program demonstrating escape sequence used in single quoted string.

```
<?php
$dir='C:\\MyFolder\\MyInfo';
$name='O\'Brien';
echo "$dir $name";
?>
```

Output: C:\MyFolder\MyInfo O'Brien

- To compare two strings for equality, use the == comparison operator. The `is_string()` function is used to check whether the given value is a string or not.

4. Boolean:

- Boolean data type allows only one of two possible values. It has been named after George Boole who defined an algebraic system of logic.

- The following values all evaluate to false.
  a) The keyword false.
  b) The integer 0.
  c) The empty string (" ") and the string "0".
  d) The floating point value 0.0.
  e) An object with no values.
  f) An array with zero elements.
  g) The NULL value.
- The value which is not false is true (including resource values).
- The `is_bool()` function is used to check whether a value is a boolean or not.

Collection (compound) data types:

1. Arrays :

- Array is a compound data type which collects group of values. An individual element in an array is referred by using an index, which is most generally an integer or it can be a string also in PHP.
- For example,

```
1) $creator[0]="James Gosling";
   $creator[1]="Guido van Rossum";
   $creator[2]="Rasmus Lerdorf";
2) $creator['Java']="James Gosling";
   $creator['Python']="Guido van Rossum";
   $creator['PHP']="Rasmus Lerdorf";
```

- In the above two examples, two separate arrays are created. Alternatively, the same arrays can be created as follows:

```
1) $creator=array('James Gosling','Guido van Rossum','Rasmus Lerdorf');
2) $creator=array('Java'=>'James Gosling', 'Python'=>'Guido van Rossum',
   'PHP'=>'Rasmus Lerdorf');
```

- Foreach loop is a special control structure to traverse through arrays in PHP. For example,

```
foreach($creator as $name)
{
    echo "$name<br>";
}
```

The `is_array()` function is used to check whether a value is an array or not.

2. Objects:

- Object is a compound data type in PHP. A class is a framework defined for creating an object. Object is an instance of a class.
- PHP supports objected oriented programming. Due to this, it becomes easier to maintain, debug the code. It also helps us to reuse the code in building applications.
- Objects are created by using new keyword and individual properties and methods of the object can be accessed using the -> notation.

Program 1.4: Write a PHP program to define a class and access its properties through an object.

```
<?php
class creator
{
    public $name=' ';
    function name($new_name=NULL)
    {
        if(!is_null($new_name))
        {
            $this->name=$new_name;
        }
    return $this->name;
    }
}
$jg=new creator;
$jg->name('James Gosling');
printf("Creator of Java : %s<br>",$jg->name);
$gr=new creator;
$gr->name('Guido van Rossum');
printf("Creator of Python : %s<br>",$gr->name);
?>
```

Output :

Creator of Java : James Gosling

Creator of Python : Guido van Rossum

- The `is_object( )` function is used to check whether a value is an object or not.

Special data types:

1. **Resources :**

- Resource is a special data type in PHP. We can open multiple files in a PHP script, the fopen function returns resource in the form of file pointer (handle) on success.
- Similarly, we can open multiple databases in a PHP script, the pg_connect function returns connection resource on success. In order to browse through a directory using PHP, we need to open the directory using opendir() function.
- The opendir() function returns either a resource in the form of directory handle on success or FALSE on failure. When there are no more references to the resource, it is automatically released by the garbage collector.

For Example:

(1)
```
<?php
$fp=fopen("emp.dat","r") or die("Please check the file name");
var_dump($fp);
fclose($fp);
?>
```

Output : resource(3) of type (stream)

(2)
```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca
user=postgres password=root") or die("Cannot Connect");
var_dump($db);
pg_close($db);
?>
```

Output : resource(2) of type (pgsql link)

- The is_resource( ) function is used to check whether a value is really a resource or not.

2. **NULL:** NULL is a special data type in PHP. There is only one value for this data type, which is of type NULL. The keyword NULL is case-insensitive.

Program 1.5: Write a PHP program to test whether a variable is having null value or not.

```
<?php
$a=null;
if(is_null($a))
   echo "\$a is null";
else
   echo "\$a is not null";
?>
```

Output : $a is null

- The is_null( ) function is used to check whether a value is null or not.

1.5.2 Variables

- The conventions for naming a variable are as follows:
  1. Variable names in PHP always begin with a dollar sign ($) and are case sensitive.
  2. Variable names do not start with a number.
  3. It must start with an alphabet or underscore.
  4. Variable name does not have any spaces.
  5. Variable names can be of any length.
- In PHP, there is no type checking on variables. Also there is no syntax for declaring variables in PHP. A variable's data type is identified by the value assigned to that variable.

  For Example:

```
$a=11;  //integer
$a='Good Day'; //string
$a=array(11,12,12); //array
```

1.5.2.1 Variable Variables

- With the use of variable variables, a variable name can be set and used at runtime (dynamic).
- We can reference the value of a variable whose name is stored in another variable by adding an additional dollar sign ($) before the variable reference.
- In simple words, variable variables are nothing but variables that hold the name of another variable.

  For Example,

```
<?php
    $what='Hello';
    $$what='PHP';
    echo "$what  ${$what}";
?>
```

- In the above example, two variables are defined. The first variable is `$what` with the contents `'Hello'` and `$Hello` with the contents `'PHP'`. In PHP, writing `$$what` inside double quotes would have generated the output as : `Hello $Hello`. To fix the problem, you have to use curly braces as shown here: `${$what}`.
- The equivalent code for the above example is shown below.

```
<?php
  $what='Hello';
  $Hello='PHP';
  echo "$what  $Hello";
?>
```

Output of both the codes is: `Hello PHP`

1.5.2.2 Variable References

- To create variable aliases (nick name or alternative name), PHP uses references. For Example, `$second=&$first;`
- Now `$second` is an alias for the variable `$first`. The old value of `$second` is now disappeared. `$second` is now an alternative name to access the value which is stored in `$first`. Unsetting a variable which is aliased does not have impact on alternative names for that variable's value.

Program 1.6: Write a PHP program to demonstrate the concept of variable references.

```
<?php
$first="chocolate";
$second=&$first;
unset($first);
print($second);
?>
```

Output: chocolate

- Function can return values by reference.

1.5.2.3 Variable Scope

- PHP has four types of variable scope, namely:
  1. Local
  2. Global
  3. Static
  4. Function parameters.

1. **Local Scope:** Any variable declared inside a function body is known to be local to that function. It is inaccessible outside that function. Variables which have been defined outside a function are inaccessible inside the body of a function.

Program 1.7: Write a PHP program to demonstrate the concept of local variable.

```
<?php
function increment_value()
{
  $value=5;
  $value++;
}
$value=10;
increment_value();
echo "$value";
?>
```

Output: 10

- The variable $value is visible only inside the function. Though the function increments the local $value variable, its scope is limited only within that function. Hence, the global $value variable's value is printed as an output.

2. **Global Scope:** Any variable declared outside a function is considered to have a global scope. But, the global variable is not directly accessible inside a function. To access a global variable inside a function, use the global keyword before the variable when declaring a variable.

Program 1.8: Write a PHP program to demonstrate the concept of global variable (use global keyword).

```
<?php
function increment_value( )
{
    global $value;
    $value++;
}
$value=10;
increment_value( );
echo "$value";
?>
```

Output: 11

- Another way is to use $GLOBALS array, which is shown in the next example.

Program 1.9: Write a PHP program to demonstrate the concept of global variable (use $GLOBALS array).

```
<?php
function increment_value( )
{
    $GLOBALS['value']++;
}
$value=10;
increment_value( );
echo "$value";
?>
```

Output: 11

3. **Static Variables:** Static variable's value is preserved between calls to a function. The scope of static variable is only within the function in which it is defined.

Program 1.10: Write a PHP program to demonstrate the concept of static variable.

```
<?php
function increment_value( )
{
    static $value=0;
    $value++;
    echo "<br>Static variable's value is $value";
    global $value;
    echo "<br>Global variable's value is $value";
}
$value=9;
increment_value( );
increment_value( );
?>
```

Output:

Static variable's value is 1
Global variable's value is 9
Static variable's value is 2
Global variable's value is 9

4. **Function parameters:** The scope of function parameters is local, which means the function parameters are accessible only inside the function. Function parameters are always declared after the function name and inside round brackets.

Program 1.11: Write a PHP program to demonstrate the concept of function parameters.

```
<?php
function disp_even_odd($num)
{
    $rem=$num%2;
    return $rem;
}
$r=disp_even_odd(11);
if($r==0)
echo "Even";
else
echo "Odd";
?>
```

Output: Odd

1.5.3 Expressions and Operators

- An expression is one of the important building blocks of PHP. The simplest forms of expressions are variables and literal values. You can build complex expressions by using simple expressions and operators.

  For Example:

  (1) $a=11; In this expression the assignment evaluates to 11.

  (2) $a+=1; This expression can also be written as $a=$a+1; In this expression the value of $a is taken and 3 is added to it and it is assigned back to $a.

Number of Operands:

- The operators are classified based on the number of operands (values) on which they operate on. In PHP there are three different types of operators. These are unary (single value), binary (two values) and ternary (three values) operators.

Arithmetic Operators:

- Arithmetic negation (–) and arithmetic assertion (+) are unary operators. Addition (+), Subtraction (–), Multiplication (*), Division (/), Modulo (%), Exponentiation (**) are binary operators.

Increment and Decrement Operators:

- PHP Supports Pre-increment, Post-increment, Pre-decrement, Post-decrement operators.

Table 1.4: Increment and Decrement Operators

Example	Name of operator	Result
++$value	Pre-increment	Increments $value by 1, then returns $value.
$value++	Post-increment	Returns $value, then increments $value by 1.
--$value	Pre-decrement	Decrements $value by 1, then returns $value.
$value--	Post-decrement	Returns $value, then decrements $value by 1.

- A simple program which illustrates the working of these operators is shown below.

Program 1.12: Write a PHP program to demonstrate both the pre and post increment and decrement operators.

```
<?php
$value=10;
echo "Pre-increment of $value is ".++$value;
$value=10;
echo "<br>Post-increment of $value is ".$value++;
$value=10;
echo "<br>Pre-decrement of $value is ".--$value;
$value=10;
echo "<br>Post-decrement of $value is ".$value--;
?>
```

Output :

Pre-increment of 10 is 11
Post-increment of 10 is 10
Pre-decrement of 10 is 9
Post-decrement of 10 is 10

Comparison Operators:

- Comparison operators compare two values. Based on the comparison, the result is either true or false. In PHP, comparison operators can handle operands of different types. The following table gives an idea about this.

Table 1.5: Result of Comparison between two operands

First Operand	Second Operand	Result of comparison
Number	Number	Numeric
String (completely numeric)	String (completely numeric)	Numeric
String (completely numeric)	String (not completely numeric)	Numeric
String (completely numeric)	Number	Numeric
String (not completely numeric)	String(not completely numeric)	Lexicographic (dictionary order)
String (not completely numeric)	Number	Lexicographic (dictionary order)

The comparison operators are listed as follows.

Table 1.6: Comparison Operators

Comparison operator	Example	Description
Equal (==)	$m==$n	Returns TRUE if both operands are equal, otherwise returns FALSE.
Identical (===)	$m===$n	Returns TRUE if both operands are equal and of the same data type, otherwise returns FALSE.
Not Equal (!= or <>)	$m!=$n $m<>$n	Returns TRUE if both operands are not equal, otherwise returns FALSE.
Not Identical (!==)	$m!==$n	Returns TRUE if both operands are not equal or they are not of the same data type, otherwise returns FALSE.

Comparison operator	Example	Description
Less than (<)	$m<$n	Returns TRUE if $m(left operand) is less than $n(right operand), otherwise returns FALSE.
Greater than (>)	$m>$n	Returns TRUE if $m(left operand) is greater than $n(right operand), otherwise returns FALSE.
Less than or equal to (<=)	$m<=$n	Returns TRUE if $m(left operand) is less than or equal to $n(right operand), otherwise returns FALSE.
Greater than or equal to (>=)	$m>=$n	Returns TRUE if $m(left operand) is greater than or equal to $n(right operand), otherwise returns FALSE.

Assignment Operators:

- The simplest assignment operator assigns a value or value of an expression to a variable. The left operand is a variable while the right operand can be literal, expression or a variable.

Program 1.13: Write a PHP program to demonstrate the assignment operator.

```
<?php
$m=1;
$n=$m+10;
echo "$n";
?>
```

Output : 11

- In the above example, $n is left operand which has been assigned the expression as the right operand. The expression evaluates to 11 which becomes the value of $n.
- In addition to this, there are some binary operators which are combined with basic assignment operator. These are : Plus-equals(+=), Minus-equals(–=), Multiply-equals(*=), Divide-equals(/=), Modulus-equals(%=), Bitwise-AND-equals(&=), Bitwise-OR-equals(|=), Bitwise-XOR-equals(^=), Concatenate-equals(.=).

Program 1.14: Write a PHP program to demonstrate the concept of assignment operator combined with the arithmetic operator.

```
<?php
$n=1;
$n+=10;    //equivalentto $n=$n+10;
echo "$n";
?>
```

Output : 11

Bitwise Operators:

- Bitwise operators carry out operations on the binary representation of the operands. The following table illustrates the working of bitwise operators in PHP.

Table 1.7: Bitwise Operators

Operator Name	Example	Description
Bitwise And (&)	$m & $n	Bits that are set (1) in both $m and $n are set as 1 in the result, otherwise set as 0 (zero).
Bitwise Or (|)	$m | $n	Bits that are set (1) in either $m or $n are set as 1 in the result, otherwise set as 0 (zero).
Bitwise Not (~)	~$m	Bits that are set (1) in $m are changed to 0 (zero) and vice versa.
Bitwise XOR (^)	$m ^ $n	If either of the bits in the pair, but not both, is 1, the resulting bit is 1, otherwise the resulting bit is 0 (zero).
Left Shift (<<)	$m << $n	Shift the bits of $m $n steps to the left(each step means "multiply by two").
Right Shift (>>)	$m >> $n	Shift the bits of $m $n steps to right(each step means "divide by two").

Logical Operators:

- Logical operators perform logical operations on TRUE and FALSE values. Hence, values which are used with a logical operator are first converted into boolean value before being evaluated.
- For example, for numerical values zero is treated as FALSE and other values are treated as TRUE. For string values, empty string is treated as FALSE and nonempty string is treated as TRUE. The following table lists the logical operators in PHP.

Table 1.8: Logical Operators

Operator Name	Example	Description
Logical AND (&&, and)	$m and $n	Returns TRUE if both $m and $n are TRUE, otherwise FALSE.
Logical OR (||, or)	$m or $n	Returns TRUE if either $m or $n is TRUE, otherwise FALSE.
Logical Negation (!)	!$m	Returns TRUE if $m is FALSE, and FALSE if $m is TRUE.
Logical XOR (xor)	$m xor $n	Returns TRUE if either $m or $n is TRUE, but not both, otherwise FALSE.

- There are two different variations of "and" and "or" operators. The two variations operate in the same manner, but they differ in operator precedence.
- If the first operand evaluates to FALSE, the logical AND operator is aware that the final result is going to be FALSE, so the second operand is not evaluated (bypassed). This is known as short-circuiting in PHP. Logical OR operator is also short-circuited. If the first operand evaluates to TRUE, the second operand is not evaluated (bypassed).

Other Miscellaneous Operators:

- **Error Suppression (@):** When this operator is added before an expression, any error messages that might be generated by that expression will be ignored.
- **Execution (` `):** PHP has execution operator, backticks(` `). The backticks is not similar to single quotes. PHP executes the contents written within backticks as a shell command.
- **Conditional (?:):** It is ternary operator(three operands).

Program 1.15: Write a PHP program to demonstrate the use of conditional (ternary) operator.

```
<?php
$m=11;
$result=($m==11) ? "yes" : "no";
echo "$result";
?>
```

Output: yes

1.5.4 Precedence of operators and Associativity

- Precedence of operators specifies in what order the operators get executed. For example, multiplication and division have higher precedence than addition and subtraction.

Program 1.16: Write a PHP program to demonstrate the precedence of operators.

```
<?php
$a=3*5+6;
echo "$a";
?>
```

Output: 21

- In the above example, multiplication has higher precedence as compared to addition. But, you can bypass the precedence rule by enclosing the expression in parentheses.

Program 1.17: Write a PHP program to demonstrate the use of parenthesis to bypass the precedence rule.

```
<?php
$a=3*(5+6);
echo "$a";
?>
```

Output : 33

- In this example, as 5+6 is enclosed in parentheses, it has higher precedence than multiplication.
- The following table lists the operators with their precedence. The operators are listed in the **highest to lowest** precedence order.

Table 1.9: Precedence of operators and Associativity

Precedence	Operator	Associativity	Operator description
Highest	clone, new	Not Applicable	clone, new operators
To	** (in PHP 5.6 onwards)	R to L	Exponentiation
Lowest (Top to Bottom)	++, --, ~, (int),(float),(string), (array),(object),(bool),@	Not Applicable	Types and increment/ decrement operators
	Instanceof	L to R	Types
	!	Not Applicable	Logical Negation
	*, /, %	L to R	Arithmetic Operators
	+, -, .	L to R	Arithmetic and String concatenation operators
	<<, >>	L to R	Bitwise Operators
	<, <=, >, >=	Non associative	Comparison Operators
	==, !=, ===, !==, <>, <=>	Non associative	Comparison Operators
	&	L to R	Bitwise and References operator
	^	L to R	Bitwise XOR Operator
	|	L to R	Bitwise OR Operator

Precedence	Operator	Associativity	Operator description
	&&	L to R	Logical AND Operator
	||	L to R	Logical OR Operator
	?? (in PHP 7.0)	R to L	Null coalescing
	? :	L to R	Ternary (Conditional) Operator
	=, +=, -=, *=, **=, /=, .=, %=, &=, |=, ^=, <<=, >>=, ??=	R to L	Assignment Operators
	yield from	Not applicable	yield form
	Yield	Not applicable	Yield
	Print	Not applicable	print
	And	L to R	Logical AND Operator
	XOR	L to R	Logical XOR Operator
	OR	L to R	Logical OR Operator

Associativity of operators:

- It defines the sequence (direction) in which the operators having same precedence are evaluated. From among the arithmetic operators, division and multiplication are having associativity as L to R (left-associative).

1.5.5 Type Juggling (Implicit) and Type Casting (Explicit)

Type Juggling (Implicit):

- In PHP, there is no explicit data type declaration for variable. In PHP, conversion of one data type to another is done implicitly as per the necessity, which is known as type juggling.

Program 1.18: Write a PHP program to demonstrate type juggling.

```
<?php
$a="10 mangoes";
$b=$a-1;                //$b is integer having value 9.
echo "$b";
?>
```

Output: 9

Type Casting(Explicit):

- Type casting in PHP is similar as in C language. The data type to which you want to cast is written in round brackets before the variable name which you want to cast.

Program 1.19: Write a PHP program to demonstrate type casting.

```
<?php
$a=11;                  //$a is an integer
$b=(float) $a;          //$b is float
printf("%f", $b);
?>
```

Output : 11.000000

- The following table shows the casts allowed in PHP.

Table 1.10: Type casting in PHP

Keyword	Description
(int), (integer)	Convert to integer
(bool), (boolean)	Convert to boolean
(float), (double), (real)	Convert to float
(string)	Convert to string
(array)	Convert to array
(object)	Convert to object
(unset)	Convert to NULL

1.5.6 Control Statements

- The control statements define the flow of execution of the program. Control statements are of different types. Some control statements are conditional (decision making), while some are loop based (repeated execution).

Conditional Statements:

(i) if statement: The if statement is conditional, which allows you to make decision and execute statements depending on the condition mentioned.

Syntax :

```
if(condition)
{
    //The statements in this block are executed if condition is true
}
```

Alternate Syntax :

```
if(condition) :
    //The statements in this block are executed if condition is true
endif;
```

(ii) if else statement: The if statement is enhanced by adding else statement to it. Whenever the condition mentioned in if statement evaluates to FALSE, the code written in the else section is executed.

Syntax :

```
if(condition)
{
    //The statements in this block are executed if condition is true.
}
else
{
    //The statements in this block are executed if condition is false.
}
```

Alternate Syntax :

```
if(condition):
    //The statements in this block are executed if condition is true.
else:
    //The statements in this block are executed if condition is false.
endif;
```

(iii) if elseif statement: In between if and else statement, you can add as many elseif statements as you require. If the condition in an elseif statement evaluates to true, then the rest of the code in the outer if statement is skipped.

Syntax :

```
if(condition1)
{
    //The statements in this block are executed if condition1 is true.
}
elseif(condition2)
{
    //The statements in this block are executed if condition2 is true.
}
else
{
    //The statements in this block are executed if condition1 and condition2
}
```

Alternate Syntax :

```
if(condition1):
    // The statements in this block are executed if condition1 is true.
elseif(condition2):
   //The statements in this block are executed if condition2 is true.
else:
   //The statements in this block are executed if condition1 and condition2
   are false.
endif;
```

(iv) Switch statement: There is an alternative to if-elseif-else structure: switch statement. In switch statement, you compare a single expression to a set of values. The break statement acts as a termination for the specific case. All statements in a matching case are executed upto the first break keyword. The default case works like the else statement. The default case is matched if no other case matches.

Syntax :

```
switch(variable)
{
    case value1:
                //statements
                break;
    case value2:
                //statements
                break;
                …
        default:
                //statements
                break;
}
```

Alternate Syntax :

```
switch(variable):
   case value1:
              //statements
              break;
   case value2:
              //statements
              break;
              …
   default:
          //statements
          break;
endswitch;
```

Loop Based Statements:

(i) while loop: The simplest type of loop in PHP is while loop. The while loop in PHP follows the same syntax as in C language.

Syntax :

```
while(expression)
{
    //statements
}
```

The statements in a while loop are executed repeatedly as long as the expression mentioned after while statement evaluates to TRUE. The value of expression is checked at each iteration of the while loop.

Alternate Syntax :

```
while(expression):
    //statements
endwhile;
```

(ii) do-while loop: The do-while loop is similar to while loop except one point. The differentiating point between while and do-while loop is that do-while loop executes the statements in its block at least once. This is because the expression is checked at the end of the iteration in do-while loop. On the other hand, the while loop checks the expression at the beginning of each iteration.

Syntax :

```
do
{
    //statements
}while(expression);
```

(iii) for loop: The for loop is similar to while loop, but it provides a better structure for building the loops.

Syntax :

```
for(initialization; condition; increment/decrement)
{
    //statements
}
```

Alternate Syntax :

```
for(initialization; condition; increment/decrement):
    //statements
end for;
```

(iv) foreach loop: The foreach loop is specially designed for arrays in PHP.

Syntax :

```
foreach($array_variable as $value)
{
    //statements
}
```

Alternate Syntax :

```
foreach($array_variable as $value):
    //statements
end foreach;
```

Other statements:

(i) **break:** To terminate a loop, break statement is used.

(ii) **continue:** The continue statement jumps to the next iteration of the loop skipping the remaining code of the loop.

(iii) **exit:** This function prints a user defined message and terminates the execution of the script. The die() function is an alias for exit.

(iv) **return:** The return statement returns program control to the calling module. If called inside a function, the return statement terminates the execution of the function. The return statement is also used to terminate the execution of an eval( ) function. The return statement returns from a function or from the script

1.5.7 Garbage Collection

- To manage memory, PHP uses reference counting and copy-on-write.
- **Copy-on-write:** When you assign a value of one variable to another variable, PHP does not get extra memory for a copy of the value.

  For example,

  ```
  $employee=array("ABC",40,"Assistant Professor");
  $other=$employee;  //array is not copied here.
  ```

  If we modify either copy, PHP actually allocates the memory and makes the copy.

  ```
  $employee[1]=41; //array is copied, value changed here.
  ```

  Thus, PHP delays the memory allocation and copying. This is known as copy-on-write.
- **Reference Counting:** Variable names are also called as symbols. All symbols are stored in a symbol table. Each value pointed to by a symbol table has a reference count. In the above example, after the initial assignment of array to `$employee` and `$employee` to `$other`, the reference count of the array reaches to 2. But as `$employee[1]` is changed, PHP now creates new array of `$employee` and the reference count of each of the arrays becomes 1. Reference count is dynamically changed. If a variable goes out of scope, its reference count is decreased by one. If the reference count of a value reaches zero(0), its memory is freed.

PROGRAMS

Program 1.20: Write a PHP program to perform arithmetic operation based on the operand and operator given by user.

```
<?php
$a=10;
$b=11;
$op='+';
if($op=='+')
  $c=$a+$b;
elseif($op=='-')
  $c=$a-$b;
elseif($op=='*')
  $c=$a*$b;
elseif($op=='/')
  $c=$a/$b;
echo "$c";
?>
```

Output: 21

Program 1.21: Write a PHP program to calculate area of triangle with base=3, height=4.

```
<?php
$base=3;
$height=4;
echo "base=3 units, height=4 units<br>";
echo "Area of triangle = (base * height)/2<br>";
$area=$base*$height/2;
echo "$area square units";
?>
```

Output:

Base = 3 units, height = 4 units

Area of triangle = (base * height) /2

6 square units

Program 1.22: Write a PHP program to check whether a given number is Armstrong or not.

```
<?php
$num=153;
$n=$num;
$result=0;
while($num>0)
{
   $rem=$num%10;
   $result+=$rem*$rem*$rem;
   $num=$num/10;
}
if($result==$n)
  echo "<b>$n</b> is an armstrong number";
else
  echo "$n is not an armstrong number";
?>
```

Output: 153 is an armstrong number

Program 1.23: Write a PHP program to print first eight terms of fibonacci series.

```
<?php
$f=1;
$s=1;
$n=8;
echo "First 8 terms of Fibonacci Series : ";
echo "$f  $s ";
$cnt=3;
while($cnt<=$n)
{
   $term=$f+$s;
   echo "$term ";
   $f=$s;
   $s=$term;
   $cnt++;
}
?>
```

Output: First 8 terms of Fibonacci Series : 1 1 2 3 5 8 13 21

Program 1.24: Write a PHP program to check whether a given number is even or odd.

```
<?php
$num=4;
if($num%2==0)
  echo "$num is an even number";
else
  echo "$num is an odd number";
?>
```

Output: 4 is an even number

Summary

- HTTP is one of the most popular application protocol used in the internet.
- HTTP is connectionless and stateless.
- Web Server helps to deliver the web content which can be accessed through the internet.
- Popular web servers are namely Apache HTTP Server, Nginx, Lighttpd, IIS etc.
- The official full form of the PHP is "Hypertext Preprocessor".
- Web Browser is an application which helps us to retrieve, present, traverse information resources on the www. Some examples of Web browsers are Google Chrome, Mozilla Firefox, Microsoft Edge, Opera, Safari, Microsoft Internet Explorer etc.
- Three main areas of PHP where PHP is used are : Server Side Scripting, Command-line scripting, Writing desktop applications.
- PHP is platform independent.
- PHP supports many databases such as PostgreSQL, MySQL, Oracle, Informix, Sybase, MongoDB etc.
- The lexical structure is the set of basic rules that regulate how you design programs in that specific language.
- Variable names in PHP are case sensitive and always begin with a dollar($) sign.
- Names of user-defined classes and functions, keywords are case insensitive.
- Whitespace does not play any role in a PHP program.
- Comment Styles in PHP : (i) C Style(multi-line) (ii) C++ Style(single line) (iii) Shell Style(single line)
- In PHP, we use define() function to define a constant.
- There is no explicit data type declaration for variables in PHP.
- There are eight data types in PHP.
- There are two formats of floating point numbers in PHP: regular, scientific.

- There are four ways to specify strings in PHP: single quoted string, double quoted string, Nowdoc, Heredoc.
- Variable variables are nothing but variables that hold the name of another variable.
- PHP has four types of variable scope: local, global, static, function parameters.
- In PHP, conversion of one data type to another is done implicitly as per the necessity, which is known as type juggling.
- The control statements in PHP are categorized into conditional and loop based.
- To manage memory, PHP uses reference counting and copy-on-write.

Check Your Understanding

Q.I Multiple choice questions.

1. Who invented the terms World Wide Web?
   (a) Rasmus Lerdorf (b) Tim Berners-Lee
   (c) Zeev Suraski (d) Andi Gutmans
2. Request Message Header contains Request Line and _____.
   (a) Request Headers (b) Request Message Body
   (c) Blank Line (d) none of these
3. Which of the following server enables users to log on to a host computer?
   (a) Telnet Server (b) List Server
   (c) FTP Server (d) Mail Server
4. Which of the following server moves one or more files securely between computers?
   (a) Telnet Server (b) List Server
   (c) FTP Server (d) Web Server
5. Zeev Suraski and ..... collaborated with Rasmus Lerdorf to rewrite the underlying parsing engine.
   (a) Andi Gutmans (b) Tim Berners-Lee
   (c) James Gosling (d) None of these
6. The lexical structure is the set of ..... rules that regulate how you design programs in that specific language.
   (a) advanced (b) new
   (c) basic (d) none of these
7. Which among the following is a compound data type?
   (a) integer (b) array
   (c) object (d) both array and object
8. Which of the following is not a special data type?
   (a) resource (b) null
   (c) boolean (d) none of these

9. Integers can be written in which of the following formats?
   (a) decimal (b) octal
   (c) hexadecimal (d) all of these
10. The _____ function is used to check whether the given value is a string or not.
   (a) is_string() (b) is_str()
   (c) chk_string() (d) check_string()
11. Any variable declared inside a function body is known to be ____ to that function.
   (a) static (b) local
   (c) global (d) all of these
12. The _____ variable (declared outside a function) is not directly accessible inside a function.
   (a) global (b) local
   (c) none of these
13. Which of the following operator when added before an expression ignores error messages?
   (a) @ (b) #
   (c) % (d) ^
14. Which of the following operator is non associative?
   (a) & (b) <<
   (c) <= (d) +=
15. Which of the following casting operator(s) is/are used explicitly to convert a value to integer?
   (a) (int) (b) (integer)
   (c) both (int) and (integer) (d) none of these
16. Which of the following casting operator(s) is/are used explicitly to convert a value to float?
   (a) (double) (b) (real)
   (c) (float) (d) all of these
17. Which of the following statement jumps to the next iteration of the loop skipping the remaining code of the loop?
   (a) break (b) continue
   (c) exit (d) return
18. The _____is an alias for exit.
   (a) die() (b) return
   (c) continue (d) none of these

19. PHP uses reference counting and ..... to manage memory.
    (a) variable management
    (b) copy-on-write
    (c) memory management
    (d) none of these

ANSWERS

1. (b)	2. (a)	3. (a)	4. (c)	5. (a)	6. (c)	7. (d)	8. (c)
9. (d)	10. (a)	11. (b)	12. (a)	13. (a)	14. (c)	15. (c)	16. (d)
17. (b)	18. (a)	19. (b)					

Q.II State True/False.

1. HTTP is connectionless and stateless protocol.
2. A Proxy Server works in between client and an external server to filter requests.
3. Zeev Suraski and Andi Gutmans collaborated with Tim Berners-Lee to rewrite the underlying parsing engine.
4. New Zend Engine II was introduced in PHP 5.0.0.
5. In PHP 7.0.0, spaceship(<=>) operator is introduced.
6. Command line scripting is the main target field for PHP.
7. A statement can be an assignment statement in the simple case or a loop with different exit points in a complex case.
8. Identifiers are used to name variables, constants, functions and classes.
9. Function names are case sensitive.
10. There is data type declaration in PHP.
11. The value of a variable is replaced if it is enclosed within double quotes, but this does not happen in case of single quotes.
12. An individual element in an array is referred by using an index, which is an integer or it can be a string also in PHP.
13. To access a global variable inside a function, use the global keyword before the variable when declaring a variable.
14. If the first operand evaluates to FALSE, the logical AND operator is aware that the final result is going to be FALSE, so the second operand is bypassed in PHP.
15. PHP does not execute the contents written within backticks as a shell command.
16. Precedence of operators specifies in what order the operators get executed.
17. Logical AND operator follows Left to Right associativity rule.
18. The control statements are not related with defining the flow of execution of the program.
19. When you assign a value of one variable to another variable, PHP does not get extra memory for a copy of the value.

ANSWERS

1. True	2. True	3. False	4. True	5. True	6. False	7. True	8. True
9. False	10. False	11. True	12. True	13. True	14. True	15. False	16. True
17. True	18. False	19. True					

Practice Questions

Q.I Answer the following questions in short.

1. List the different types of servers.
2. List some of the popular web servers.
3. What are the three main areas in which PHP is used?
4. What is a statement in PHP?
5. What are the different comment styles in PHP?
6. What are identifiers?
7. What are the different data types in PHP?
8. How to check the data type of a variable in PHP?
9. What are variable variables in PHP?
10. What is a static variable?
11. List the different comparison operators used in PHP.
12. State the use of execution (backticks) operator in PHP.
13. What is the difference between if-else statement and switch statement?
14. State the difference between while and do-while loop.
15. State the purpose of break statement.

Q.II Answer the following questions.

1. What are the features of HTTP?
2. Explain Web server in detail.
3. Explain the reasons why people choose PHP?
4. Explain data types in PHP in detail.
5. What are the conventions for naming a variable?
6. What are variable references?
7. Explain the different variable scopes in PHP.
8. Describe the different comparison operators used in PHP.
9. Explain how precedence and associativity affects the execution of operators.
10. Compare between for and foreach loop.
11. Explain reference counting and copy-on-write in PHP.

Q.III Define the terms

1. Web Server
2. Web Browser
3. Literal
4. Variable Interpolation
5. Short circuiting
6. Type juggling
7. Precedence of operators
8. Copy-on-write
9. Reference counting

***

2... Functions and Arrays

Objectives...

- To study about function.
- To learn Default parameters, Variable parameters, Missing parameters.
- To understand difference between Indexed Vs Associative Arrays.
- To study multidiamentional array.
- To learn traversing and sorting using array.

2.1 INTRODUCTION

- A function is a block of code written in a program to perform some specific task. They take information as parameter, execute a block of statements or perform operations on these parameters and return the result.
- Functions save on compile time no matter how many times you call them, functions are compiled only once for the page.
- PHP provides us with two major types of functions:
  - **Built-in functions:** PHP provides us with huge collection of built-in library functions. These functions are already coded and stored in form of functions.
  - **User defined functions:** Apart from the built-in functions, PHP allows users to create customized functions called as user-defined functions. Using this we can create our own packages of code and use it wherever necessary by simply calling it.

Advantages of PHP Functions:

1. **Code Reusability:** In PHP, user defined functions are defined only once and can be invoked many times, like in other programming languages.
2. **Less Code:** It saves a time of writing code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.
3. **Easy to understand:** PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

2.2 DEFINGING AND CALLING A FUNCTIONS

- Function definitions begin with the *function* keyword, followed by the function name and a list of parameters (optional) separated by comma in parentheses.
- The function name can be any string that starts with a letter or underscore followed by zero or more letters, underscores, and digits. Function names are case-insensitive.
- Curly braces then enclose the main body of the function, which can contain any legal PHP code, including variable definitions, conditional tests, loops, and output statements.
- Typically, a function returns some value.
- **To define a function, use the following syntax:**

```
function [&] function name ( [ parameter [, ... ] ] )
{
    //code to be executed
}
```

- There are three components to every function:
  - **Parameters**: Serves as inputs to the function.
  - **Return values**: The output returned by the function.
  - **Function body**: The processing code to turn inputs into outputs.
- To return a value from a function, use the return statement inside function body:
- **Syntax:** `return expr`
- When a return statement is encountered during execution, control transfers to the calling statement, and the evaluated results of `expr` will be returned as the value of the function.
- If you define your function with the optional ampersand before the name, the function returns a reference to data rather than a copy of the data.

2.2.1 Function without Parameters

- The function in PHP can be written without any parameters.
- Let's take a look at a simple function.

Program 2.1: Write a PHP Program to illustrate the concept of defining a function without parameter.

```
<?php
function message(){
echo "Good Morning";   //function body
}
message( );// function call
?>
```

Ouput:

Good Morning

2.2.2 Function with Parameters

- Functions can take an arbitrary number of arguments by declaring them in the function definition. There are two different ways of passing parameters to a function- by value and by reference.

Passing Parameters by Value:

- In most cases, you pass parameters to a function by value. The argument is any valid expression. An expression is evaluated, and the resulting value is assigned to the appropriate formal parameter of the function.

Program 2.2: Write a PHP program to define addition( ) function which takes two numbers as parameters and display their addition.

```
<?php
function addition ($num1, $num2)   // function along with two parameters
{
   $sum = $num1 + $num2;        //function body
   echo "The Addition is $sum";
}
$x=10;
$y=20;
addition($x, $y);     //or addition(10, 20); function call
?>
```

Output: The Addition is 30

- In above example, $x will be assigned to $num1 and $y will be assigned to $num2 respectively. These values will be used to compute the result of addition and echo statement will display the result.

Passing Parameters by Reference:

- Passing parameters by reference allows direct access to a variable. To be passed by reference, the argument must be a variable; you indicate that a particular argument of a function will be passed by reference by preceding the formal parameter in the parameter list with an ampersand (&) character.

Program 2.3: Write a PHP program to illustrate the concept of passing parameters to a function by reference.

```
<?php
function update(&$value)
{
  $value++;
}
$count=1;
update($count);
echo "The Count is ". $count;
?>
```

Output: The Count is 2

- Because the function's $value parameter is passed by reference, the actual value of $count, rather than a copy of that value, is modified by the function. Before, we had to return the incremented value, but now we change the caller's $count variable to be incremented by 1.

Program 2.4: Write a PHP program to display concatenated strings using parameter passing by reference.

```
<?php
function strconcat(&$str2)
{
    $str2 .= 'By Reference';
}
$str1 = 'Parameter Passing ';
strconcat($str1);
echo $str1;
?>
```

Output:

Parameter Passing By Reference

- In this example, variable $str1 is passed to the strconcat( ) function where it is concatenated with 'By Reference' string. Here, printing $str1 variable results 'Parameter Passing By Reference'. It is because changes are done in the actual variable $str1.
- Note: A parameter that is being passed by reference can only be a variable.

2.2.3 Function Returning the Result

Return a Value:

- You can have a function explicitly return a single value in PHP like the result of a calculation, to the statement that called it. This is accomplished by using a return statement inside the function, as in the following example.

Program 2.5: Write a PHP program to define addition ( ) function which takes two numbers as parameters and return their addition.

```
<? php
function addition ($num1, $num2)
{
    $sum = $num1 + $num2;
    return  $sum;
}
echo "The Addition is ". addition(4, 3);          // function call
?>
```

Output:

The Addition is 7

- The returning value may be of any data type including the arrays and objects.

Return Multiple Values:

- You can return multiple values from a function, by placing them all in an array and returning the array. The next example illustrates this concept.

Program 2.6: Write a PHP program to illustrate the concept of returning multiple values from a function.

```
<?php
function cumsum($a,$n)
{
  $sum=array(0,0,0,0,0);
  $j=0;
  $sum[0]=$a[0];
  for($i=1;$i<$n;$i++,$j++)
     $sum[$j+1]= $sum[$j]+$a[$i];
  return $sum;
}
$n=5;
$a=array(1,2,3,4,5);
$b=cumsum($a,$n);  //$sum array is returned and stored in $b
 print_r($b);
?>
```

Output:

Array ([0] => 1 [1] => 3 [2] => 6 [3] => 10 [4] => 15)

Return a Reference:

- If you define a function with the optional ampersand before the name, the function returns a reference to data rather than a copy of the data.

Program 2.7: Write a PHP program to illustrate the concept of returning a reference from the function.

```
<?php
function & findmax(&$m,&$n)
{
   if ($m > $n)
      return $m;
      return $n;
```

```
}
$no1=5;
$no2=10;

echo " Number 1: $no1 <BR> Number 2: $no2";
//returned reference will be stored in $max variable
$max =  & findmax($no1,$no2);
$max=-1;   // maximum will be set to -1
echo "<BR>Numbers after function call:<BR> Number 1: $no1 <BR> Number 2:
$no2";
?>
```

Output:

Number 1: 5
Number 2: 10
Numbers after function call:
Number 1: 5
Number 2: –1

Here, parameters $no1 and $no2 are passed by reference into $m and $n variables respectively. Both variables are compared and maximum value is returned by a reference, which is stored in $max, so finally $max is assigned to –1. In this case $no2's reference variable $n would be returned and is set to –1, so in output $no2= –1.

2.3 DEFAULT PARAMETERS

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

Program 2.8: Write a PHP program to illustrate the concept of default parameter.

```
<?php
function addition ($num1, $num2=5)
{
    $sum = $num1 + $num2;
    echo "The Addition of $num1 and $num2 is $sum";
}
$x=10;
$y=20;
addition($x, $y);
addition($x);    //function call where default argument will be used
?>
```

Output:

The Addition of 10 and 20 is 30
The Addition of 10 and 5 is 15

- In `addition($x, $y)` function call both parameters are passed, so `$num1` will take value of `$x` and `$num2` will take value of `$y` respectively. But in second call to `addition($x)` function, only one parameter is passed which will be assigned to `$num1` and for second parameter `$num2`, it will take default value for `$num2` as 5.
- The value assigned to a parameter as a default value cannot be a complex expression; it can only be a constant.
- A function may have any number of parameters with default values. However, they must be listed after all the parameters that do not have default values i.e. only trailing arguments can have default values.

2.4 VARIABLE PARAMETERS AND MISSING PARAMETERS

Variable Parameters:

- PHP allows to define functions with variable length argument lists, where the number of arguments can change with each invocation of the function.
- PHP provides three functions to retrieve the parameters passed to it:
  - `func_get_args( )`: It returns an array of all parameters provided to the function.
    `$array = func_get_args( );`
  - `func_num_args( )`: It returns the number of parameters provided to the function.
    `$count = func_num_args( );`
  - `func_get_arg( )`: It returns a specific position argument from the parameter list.
    `$value = func_get_arg(argument_number);`

Program 2.9: Write a PHP program to illustrate the concept of variable number of parameters to a function.

```
<?php
function addition( )
{
 if(func_num_args( ) == 0)
 {
      return false;
 }
 else
 {
      for($i = 0; $i < func_num_args( ); $i++)
      {
          $sum+= func_get_arg($i);
      }
      return $sum;
```

```
}
}
echo "Addition of 1, 2 and 3 is ".addition(1, 2, 3)."<br>";
echo "Addition of 4, 6, 8 and 10 is ".addition(4, 6, 8, 10);
echo addition();
?>
```

Output:

Addition of 1, 2 and 3 is 6

Addition of 4, 6, 8 and 10 is 28

- Here the `addition( )` function takes any number of arguments. It loops over those arguments and returns the total of all the parameter values. If no parameters are given, it returns false.
- The result of any of these functions cannot directly be used as a parameter to another function. To use the result of one of these functions as a parameter, first set a variable to the result of the function then use that in the function call. The following expression will not work:

```
funcall(func_num_args( ));
```

- Instead, use:

```
$count = func_num_args( );
funcall($count);
```

Missing Parameters:

- In PHP, when you call a function, you can pass any number of arguments to the function. Any parameters the function expects that are not passed to it remain unset, and a warning is issued for each of them:

Program 2.10: Write a PHP program to illustrate the concept of missing parameters to a function.

```
<?php
function MissingFunPar( $x, $y )
{
    if (isset($x)) { echo " x is $x\n"; }
    if (isset($y)) { echo " y is $y\n"; }
}
echo "For function call MissingFunPar (10, 20):\n";
MissingFunPar (10, 20);
echo "For function call MissingFunPar (10):\n";
MissingFunPar (10);
?>
```

Output:

For function call MissingFunPar(10, 20):

x is 10

y is 20

For function call `MissingFunPar(10)`:

Warning: Missing argument 2 for `MissingFunPar( )`

`in /path/to/script.php on line 10`

x is 10

2.5 VARIABLE FUNCTION AND ANONYMOUS FUNCTION

Variable Function:

- A function name can be stored in a variable, so we can write parenthesis after a variable name to call that function.
- Parameters also can be passed to a variable function like any other function
- **Syntax:** `$variable();`

Program 2.11: Write a PHP program to illustrate the concept of variable function.

```
<?php
function addition($a,$b)
{
   return ($a+$b);
}
function subtract($a,$b)
{
   return ($a-$b);
}
$fun="addition";
echo "<br>Addition of 20 and 5 is: ".$fun(20,5);

$fun="subtract";
echo "<br>Subtraction of 20 and 5 is: ".$fun(20,5);
?>
```

Output:

Addition of 20 and 5 is: 25

Subtraction of 20 and 5 is: 15

Anonymous Function:

- PHP allows to define a function without any name are called as Anonymous function. It is also called as **closure**.
- It is assigned directly to a variable and can be called by the variable name or can be passed as parameter to any function.
- These functions are very useful because that are throw away functions i.e. you can use them once. After the use, they will not be in the global scope. This will save memory requirement of the program.
- It terminates with semicolon(;).

Program 2.12: Write a PHP program to illustrate the concept of anonymous function.

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

Output:

Hello

- Parameters can be passed to an anonymous function.

Program 2.13: Write a PHP program to illustrate the concept of anonymous function with parameters.

```
<?php
  $add=function($a,$b) { return $a+$b ;};
  echo "<br>Addition of 20 and 10: ".$add(20,10);
?>
```

Output:

Addition of 20 and 10: 30

- Like any other functions, anonymous functions can use variables declared in the global scope.

Program 2.14: Write a PHP program to illustrate how anonymous functions can use variables declared in the global scope.

```
<?php
$c="Atharv";
$greeting=function() use ($c)
{
      echo "<br>Hello $c";
};
$greeting();
?>
```

Output:

Hello Atharv

- Anonymous functions can be stored in an array and invoke a particular function from an array using subscript [] notation.

Program 2.15: Write a PHP program to illustrate how anonymous functions can be stored in an array and how to invoke a particular function from an array.

```
<?php
$arr=array(function(){ echo "Function 1 is invoked";},
           function(){ echo "Function 2 is invoked";},
           function(){ echo "Function 3 is invoked";}
           );
$arr[1]();   //call to a function
?>
```

Output:

Function 2 is invoked

Differences between Anonymous function and Normal function:

Table 2.1: Differences between Anonymous function and Normal function

Sr. No.	Anonymous function	Normal function
1.	There is no name to a function.	Every function has a name.
2.	Function definition end with semicolon (;)	Function definition doesn't end with semicolon (;)
3.	It is assigned to a variable and then called using variable name followed by parenthesis.	It is called using function name followed by parenthsis.
4.	Function will not be in the global scope.	Function will remain in the global scope.

2.6 INDEXED VS ASSOCIATIVE ARRAYS

Indexed Array:

- This type of array can be used to store elements of any data type.
- An index is used to access any element from an array which will be always a number. By default, the index starts at zero.
- PHP rules for naming arrays is similar as like regular variables, as they proceed with $ and should start with letter or underscore.

Associative Array:

- PHP allows you to associate name/label with each array element using => symbol.
- There are two ways to create associative arrays, the first method in which all elements are assigned as index/key => value pairs and are separated with commas.
- **Example:**

```
$weight=array('Suresh'=>'55','Ramesh'=>'60','Rajesh'=>'70');
```

- The another way to create array is to set values individually using index notation.
- **Example:**

```
$age['Suresh'] = "55";
$age['Ramesh'] = "60";
$age['Rajesh'] = "70";
```

2.7 IDENTIFYING ELEMENTS OF AN ARRAY

- To access a value from an array in PHP script, use the array name and element's key or index.
- The key can be either an integer or a string.
- **Example:**

```
$cities=array('Pune', 'Mumbai', 'Nashik', 'Kolhapur');
```

With `$cities` indexed array, `$cities[1]` will refer to an array element 'Mumbai'.
With `$age` associative array, `$age['Ramesh']` will refer to an array element '60'.

- String index values are equivalent to integer numbers i.e. `$cities[2]` is same as `$cities['2']`.

2.8 STORING DATA IN ARRAYS

- Elements can be stored in an array by mentioning its index and value to be stored in an array.
- If an array is initially empty then by storing elements in an array will create the array.
- There are two ways to create arrays in PHP, the first method in which all elements are assigned at once and are separated with commas.
- **Example of Indexed Array:**

```
$cities = array('Pune', 'Mumbai', 'Nashik', 'Kolhapur');
```

OR

```
$cities = ['Pune', 'Mumbai', 'Nashik', 'Kolhapur');
```

- **Example of Associative Array:**

```
$weight=array('Suresh'=>'55','Ramesh'=>'60','Rajesh'=>'70');
```

OR

```
$weight=['Suresh'=>'55','Ramesh'=>'60','Rajesh'=>'70'];
```

- The second way to create array is to set values individually using index notation.
- **Example of Indexed Array:**

```
$cities = array();
$cities[0] = 'Pune';
$cities[1] = 'Mumbai';
$cities[2] = 'Nashik';
$cities[3] = 'Kolhapur';
echo $cities[1];    // prints Mumbai
```

- **Example of Associative Array:**

```
$age = array();
$age['Suresh'] = "55";
$age['Ramesh'] = "60";
$age['Rajesh'] = "70";
```

The statement: echo `$age[2];` will print 70.

- With associative array, it is not necessary to mention keys of all elements. If the key is mentioned for first element as '2' then further elements will have subsequent values for keys.
- **Example:**

```
$days = array('2'=>'Mon','Tue','Wed','Thur','Fri','Sat','Sun');
```

Here index of `'Tue'` is 3, `'Wed'` is 4 etc.

- If the key is mentioned for first element as non-numeric string then further elements will have integer indices starting with 0(zero)..
- **Example:**

```
$days= array('one'=>'Mon','Tue','Wed','Thur','Fri','Sat','Sun');
```

This is same as:

```
$days= array('one'=>'Mon',0=>'Tue',1=>'Wed',2=>'Thur',3=>'Fri',4=>'Sat',5=>'Sun');
```

2.8.1 Add values at end

- You can add element in the existing array after the last element without knowing the index of the last element.
- **Example:**

```
$no = array(10,20,30,40);
$no[] = 50;
```

Here value of `$no[4]` is 50. It assumes array index values are numbers.

- With associative array, if element is added and index is not mentioned then PHP will assign numeric index.
- **Example:**

```
$a = array('one'=>1, 'two'=>2, 'three'=>3, 'four'=>4);
$a[]=5;
```

Here value of `$a[0]` is 5. It assigns index values starting with 0.

2.8.2 Range of Values

- **range( ):** This function creates an array of consecutive integer or character values between the two values you pass to it as arguments.
- **Syntax:** `array range(start,end)`

Program 2.16: Write a PHP program to illustrate the use of range() function.

```
<?php
$a1=range(2,10);
echo "<BR>Array from 2 to 10:<BR>";
print_r($a1);

$a2=range('b','e');
echo "<BR>Array from b to e:<BR>";
print_r($a2);

$a3=range('aaa','ddd');
echo "<BR>Array from aaa to ddd:<BR>";
print_r($a3);
?>
```

Output:

Array from 2 to 10:
Array ([0] => 2 [1] => 3 [2] => 4 [3] => 5 [4] => 6 [5] => 7 [6] => 8 [7] => 9 [8] => 10)
Array from b to e:
Array ([0] => b [1] => c [2] => d [3] => e)
Array from aaa to ddd:
Array ([0] => a [1] => b [2] => c [3] => d)

2.8.3 Size of an Array

- **count( ) and sizeof( ):** The count( ) and sizeof( ) functions are identical in use and effect. They return the number of elements in the array.
- **Syntax:**

```
integer count(array);
integer sizeof(array);
```

Program 2.17: Write a PHP program to illustrate the use of count() and sizeof() function.

```
<?php
$a=array( );
$size = count($a);                    // $size is 0
print_r($a);
echo "<BR> Size of the array is: ".$size."<BR>";

$b = array('ABC', 'XYZ', 'PQR');      //indexed array
$size = count($b);                    // $size is 3
print_r($b);
echo "<BR> Size of the array is: ".$size."<BR>";
```

```
$c = array( 1 => 'one', 2 => 'two', 3 => 'three', 4=>'four');
//associative array
$size = sizeof($c);
print_r($c);                                    // $size is 4
echo "<BR> Size of the array is: ".$size."<BR>";
?>
```

Output:

Array ()
Size of the array is: 0
Array ([0] => ABC [1] => XYZ [2] => PQR)
Size of the array is: 3
Array ([1] => one [2] => two [3] => three [4] => four)
Size of the array is: 4

2.8.4 Padding an Array

- **array_pad():** This function is used to create an array of required size with elements initialized to some value.
- **Syntax:** `array array_pad(array, array_size, ,initial_ value);`
  - Here first argument is the array to which elements to be padded, second argument is size of array after padding of elements and third argument is the initial value of the elements which are padded.
  - If second argument is poisitive then elements with initial value will be padded at the end and if it is negative then elements with initial value will be padded at the beginning.
  - It returns the new padded array.

Program 2.18: Write a PHP program to illustrate the use of array_pad() function.

```
<?php
$a=array(1,2,3);
echo "<br>Given Array: <br>";
print_r($a);

$p1=array_pad($a,7,11);
echo "<br><br>New Array after padding(Second arg is positive): <br>";
print_r($p1);
$p2=array_pad($a,-7,0);
echo "<br><br>New Array after padding(Second arg is negative): <br>";
print_r($p2);
?>
```

Output:

Given Array:
Array ([0] => 1[1] => 2[2] => 3)

New Array after padding(Second arg is positive):
Array ([0] => 1 [1] => 2 [2] => 3 [3] => 11 [4] => 11 [5] => 11 [6] => 11)

New Array after padding(Second arg is negative):
Array ([0] => 0 [1] => 0 [2] => 0 [3] => 0 [4] => 1 [5] => 2 [6] => 3)

- Here, array $p1 have all padded elements at the end with initial value as 11 and array $p2 have all padded elements at the beginning with initial value as 0.

2.9 MULTIDIMENSIONAL ARRAYS

- It is an array which stores another array at each index rather than storing single value. In other words, multidimensional array is an array of arrays. It can be an indexed array or associative array.
- **Example:**

```
<?php
$r1 = array(11,22,33);
$r2 = array(44,55,66);
$r3 = array(77,88,99);
$mularr = array($r1,$r2,$r3);
```

- Any element can be accessed by referring its index for every dimension. For example, in case of two dimentional array, you have to mention one index for 1st dimension and another index for 2nd dimension. Every dimension's index value will start from 0 for Indexed array.
- **Example:** An element 77 can be accessed as `$mularr[2][0]`.
- Multidimensional array can also be an associative array.

Program 2.19: Write a PHP program to illustrate the concept of multidimensional associative array.

```
<?php
$r1 = array('a'=>11,'b'=>22,'c'=>33);
$r2 = array('d'=>44,'e'=>55,'f'=>66);
$r3 = array('g'=>77,'h'=>88,'i'=>99);
$mularr = array($r1,$r2,$r3);
echo "{$mularr[1]['d']}";
?>
```

Output: 44

Program 2.20: Write a PHP program to illustrate the concept of multidimensional associative array.

```
<?php
$calendar = array('weekday' => array('Monday', 'Tuesday', 'Wednesday',
'Thursday','Friday','Saturday','Sunday'),'month' => array('January',
'February', 'March', 'April','May','June'),'year' => array('2018', '2019',
'2020'));
echo "{$calendar['month'][3]}";
?>
```

Output: April

2.10 EXTRACTING MULTIPLE VALUES

- **list():** This function is used to store array values into variables.
- **Syntax:** `list($var1,$var2,...) = $array;`
- Array values will be extracted into variables in the order in which they are present in the array.
- If you have more values in the array than number of variables mentioned in the list, then extra values will be ignored.
- If you have more variables in the list than number of values in the array, then extra variables will be set to NULL.
- Two or more consecutive commas in the list() skip values in the array.

Program 2.21: Write a PHP program to illustrate the use of list() function.

```
<?php
$a=array(1,2,3);
list($x,$y,$z)=$a;
echo "<br>x=$x, y=$y, z=$z<br>";

$a=array(1,2);
list($x,$y,$z)=$a;
echo "<br>x=$x, y=$y, z=$z<br>";

$a=array(1,2,3,4,5,6,7);
list($x,$y,$z)=$a;
echo "<br>x=$x, y=$y, z=$z<br>";

list($x,,$y,,,$z)=$a;
echo "<br>x=$x, y=$y, z=$z<br>";
?>
```

Output:

x=1, y=2, z=3
x=1, y=2, z= // here z is NULL
x=1, y=2, z=3 //extra values from array are ignored
x=1, y=3, z=6

- In the statement: `list($x,$y,$z)=$a` where `$a=array(1,2)`, we have more variables in the list than number of values in the array, so `$z` is set to NULL.
- In the statement: `list($x,,$y,,,$z)=$a`, double commas skips one element and triple commas skips two elements.

2.10.1 Slicing an Array

- **array_slice( ):** This function is used to extract only a subset of the array. The array_slice( ) function returns a new array consisting of a consecutive series of values from the original array.
- **Syntax:** `array array_slice(array, offset, length)`
- The *offset* parameter identifies the initial element to copy (0 represents the first element in the array), and the *length* parameter identifies the number of values to copy. The new array has consecutive numeric keys starting at 0.

Program 2.22: Write a PHP program to illustrate the use of array_slice() function

```
<?php
$a1=array(2,3,4,7,9);
echo "<BR> Array a1: <BR>";
print_r($a1);

$slice1 = array_slice($a1, 1, 3);
echo "<BR> Array a1 after slice from 1(start position) to 3(Number of
elements) : <BR>";
print_r($slice1);

$a2 = array('String1', 'String2','String3','String4');
echo "<BR>Array a2: <BR>";;
print_r($a2);

$slice2 = array_slice($a2, 2, 2);
echo "<BR> Array a2 after slice from 2(start position) to 2(Number of
elements) : <BR>";
print_r($slice2);
?>
```

Output:

Array a1:
Array ([0] => 2 [1] => 3 [2] => 4 [3] => 7 [4] => 9)
Array a1 after slice from 1(start position) to 3(Number of elements) :
Array ([0] => 3 [1] => 4 [2] => 7)
Array a2:
Array ([0] => String1 [1] => String2 [2] => String3 [3] => String4)
Array a2 after slice from 2(start position) to 2(Number of elements) :
Array ([0] => String3 [1] => String4)

2.10.2 Spliting an Array into Chunks

- **array_chunk():** This function divides an array into smaller, evenly sized arrays.
- **Syntax:** `array= array_chunk(array, size,preserve_key);`
- The function returns an array of the smaller arrays. The third argument preserve_keys is optional and it is a boolean value that keeps the same keys in the chunk as in the original array.

Program 2.23: Write a PHP program to illustrate the use of array_chunk() function.

```
<?php
$a=array(10,20,30,40,50);
print_r($a);
echo "<br>Without Preserving Keys<br>";

$ca=array_chunk($a,2);
print_r($ca);

echo "<br>With Preserving Keys<br>";
$ca=array_chunk($a,2,true);
print_r($ca);
?>
```

Output:

Array ([0] => 10 [1] => 20 [2] => 30 [3] => 40 [4] => 50)
Without Preserving Keys
Array ([0] => Array ([0] => 10 [1] => 20)
[1] => Array ([0] => 30 [1] => 40)
[2] => Array ([0] => 50)
)
With Preserving Keys
Array ([0] => Array ([0] => 10 [1] => 20)
[1] => Array ([2] => 30 [3] => 40)
[2] => Array ([4] => 50))
)

- Here array $a is divided into smaller equal sized arrays. Each chunk contains 2 elements with starting index as 0. As there are 5 elements in an array, last chunk contains single element.
- In the statement: `array_chunk($a,2,preserve_keys)`, last argument will preserve the keys of elements in each chunk.

2.10.3 Keys and Values

- **array_keys():** This function returns an array containing only the keys from the array which is passed as parameter. Here the order of elements is considered as array's internal order.
- **Syntax:** `$keys = array_keys(array);`
- **array_values():** This function returns an array containing only the values from the array which is passed as parameter.
- **Syntax:** `$values = array_values(array);`

Program 2.24: Write a PHP program to illustrate the use of array_keys() and array_values() functions.

```
<?php
$a=array("Ritika"=>"FYBCA", "Atharv"=>"SYBCA", "Shailesh"=>"TYBCA");
 print_r($a);

 echo "<br><br>Array keys:<br>";
 $keys=array_keys($a);
 print_r($keys);

 echo "<br><br>Array values:<br>";
 $values=array_values($a);
 print_r($values);
?>
```

Output:

Array ([Ritika] => FYBCA [Atharv] => SYBCA [Shailesh] => TYBCA)

Array keys:

Array ([0] => Ritika [1] => Atharv [2] => Shailesh)

Array values:

Array ([0] => FYBCA [1] => SYBCA [2] => TYBCA)

2.10.4 Checking whether an Elements Exists

- **array_key_exists():** This function checks if an element's key exists in the array or not.
- **Syntax:** `boolean_value=array_key_exists(key, array))`
- The function returns a boolean value that indicates whether the first argument *key* is a valid key in the array given as the second argument.

Program 2.25: Write a PHP program to illustrate the use of `array_key_exists()` function.

```
<?php
$a=array("Ritika"=>"FYBCA", "Atharv"=>"SYBCA","Shailesh"=>"TYBCA");
print_r($a);
$key="Atharv";
if(array_key_exists($key, $a))
    echo "<br>Key '".$key."' found";
else
   echo "<br>Key $key not found";
$key="Rupali";
if(array_key_exists($key, $a))
   echo "<br>Key '".$key."' found";
else
   echo "<br>Key '".$key."' not found";
?>
```

Output:

Array ([Ritika] => FYBCA [Atharv] => SYBCA [Shailesh] => TYBCA)
Key 'Atharv' found
Key 'Rupali' not found

2.10.5 Removing and inserting element

- **array_splice():** This function can be used to remove elements, insert elements or replace existing elements from an array:
- **Syntax:**

```
removed_array = array_splice(array, start [, length [,
replacement_array] ]);
```

The possible operations are:

- **Remove elements:** To perform this operation, meaning of parameters is as follows:
  - This function takes an *array* as first argument whose *length* number of elements to be removed from *start* position.
  - If *length* is not mentioned then elements will be removed till the end.
  - This function returns an array of removed elements.

Program 2.26: Write a PHP program to illustrate the use of array_splice() function to delete elements from an array.

```
<?php
$a=array('a','b','c','d','e','f','g');
echo "<br> Original Array<br>";
print_r($a);

$ar=array_splice($a,2);  //remove elements from start index 2 till the end
echo "<br> Removed elements:<br>";
print_r($ar);

echo "<br> Remaining elements of array<br>";
print_r($a);
echo "<br><br>";

$a=array('a','b','c','d','e','f','g');
echo "<br> Original Array<br>";
print_r($a);

$ar=array_splice($a,2,3); );  //remove 3 elements from start index 2
echo "<br>Removed elements:<br>";
print_r($ar);

echo "<BR> Remaining elements of array<br>";
print_r($a);
?>
```

Output:

Original Array
Array ([0] => a [1] => b [2] => c [3] => d [4] => e [5] => f [6] => g)
Removed elements:
Array ([0] => c [1] => d [2] => e [3] => f [4] => g)
Remaining elements of array
Array ([0] => a [1] => b)
Original Array
Array ([0] => a [1] => b [2] => c [3] => d [4] => e [5] => f [6] => g)
Removed elements:
Array ([0] => c [1] => d [2] => e)
Remaining elements of array
Array ([0] => a [1] => b [2] => f [3] => g)

- **Replace elements:** To perform this operation, meaning of parameters is as follows:
  - This function takes an *array* as first argument whose *length* number of elements from *start* position to be deleted and insert elements from *replacement_array*.
  - The size of the *replacement_array* doesn't have to be the same as the number of elements you delete. The array grows or shrinks as needed.

Program 2.27: Write a PHP program to illustrate the use of array_splice() function to replace elements from an existing array with new array elements.

```
<?php
$a=array('a','b','c','d','e','f','g');
echo "<br><br> Original Array1<br>";
print_r($a);

$b=array('i','j','k');
echo "<br> Original Array2<br>";
print_r($b);

$ar=array_splice($a,1,2,$b);
echo "<br>Array1 after replacement<br>";
print_r($a);
?>
```

Output:

Original Array1

Array ([0] => a [1] => b [2] => c [3] => d [4] => e [5] => f [6] => g)

Original Array2

Array ([0] => i [1] => j [2] => k)

Array1 after replacement

Array ([0] => a [1] => i [2] => j [3] => k [4] => d [5] => e [6] => f [7] => g)

- Here 2 elements from start index 1 will be deleted from array $a and there $b elements are inserted. We have deleted 2 elements and inserted 3 elements.
- **Insert elements:** This function takes an array as first argument in which elements from `replacement_array` will be inserted from start position. The parameter length should be 0, so that no elements will be removed.

Program 2.28: Write a PHP program to illustrate the use of array_splice() function to insert elements from an one array into an another array.

```
<?php
$a=array('a','b','c','d','e','f','g');
echo "<br><br> Original Array1<br>";
print_r($a);

echo "<br> Original Array2<br>";
print_r($b);

$ar=array_splice($a,2,0,$b);
echo "<br>Array1 after insertion of elements from Array2<br>";
print_r($a);
echo "<BR>";
?>
```

Output:

```
Original Array1
Array ( [0] => a [1] => b [2] => c [3] => d [4] => e [5] => f [6] => g )
Original Array2
Array ( [0] => i [1] => j [2] => k )
Array1 after insertion of elements from Array2
Array ( [0] => a [1] => b [2] => i [3] => j [4] => k [5] => c [6] => d
[7] => e [8] => f [9] => g )
```

- Here from start index 2, delete 0 elements. That means no elements are deleted. Also from start index 2, elements from $b are inserted.

2.11 TRAVERSING ARRAYS

- There are different methods of traversing array elements. Also while traversing an array, different operations can be performed like adding up all elements of an array, call a function for every element of an array. The function to be used depends on the data and the task you want to perform

2.11.1 for loop

- This loop can be used when the user knows how many elements exist in an array. That is, the number of iterations is known beforehand. These types of loops are also known as entry-controlled loops.
- **Syntax:**

```
for(initialization statement; condition; increment/decrement statement)
{
    //code to be executed
}
```

- **Alternative Syntax:**

```
for(initialization statement; condition; increment/decrement statement):
//code to be executed
end for;
```

Program 2.29: Write a PHP program to illustrate the use of for loop with indexed array.

```
<?php
 $arr = array (5,10,15,20,25);
 for ($i=0;$i<sizeof($arr);$i++) {
      echo "$arr[$i] ";
 }
 echo "<br>";
 $arr = array ("Ritika", "Atharv", "Shailesh");
 for ($i=0;$i<sizeof($arr);$i++) {
      echo "$arr[$i] ";  }
?>
```

Output:

5 10 15 20 25

Ritika Atharv Shailesh

2.11.2 foreach loop

- The *foreach* statement allows you to iterate over elements in an array. For indexed array, to loop over an array, accessing the value at each key.
- **Syntax:**

```
foreach($array as $var)
{
    //code to be executed
}
```

- **Alternative Syntax:**

```
foreach($array as $var):
//code to be executed
endforeach;
```

- With a foreach loop, each time the loop runs, the current array element is assigned to a temporary variable, which can then be processed in way statements written in the loop body.
- Unlike a for loop, a foreach loop doesn't use a counter; it automatically "knows" where it is in the array, and it moves forward continuously until it reaches the end of the array, at which point it automatically halts.

Program 2.30: Write a PHP program to illustrate the use of foreach loop with indexed array.

```
<?php
  $arr = array (5,10,15,20,25);
  foreach ($arr as $val) {  //fetches one element into $val
      echo "$val\n";
  }
  echo "<br>";
  $arr = array ("Ritika", "Atharv", "Shailesh");
  foreach ($arr as $val) { //fetches one element into $val
      echo "$val\n";
  }
  ?>
```

Output:

```
5
10
15
20
25
Ritika
Atharv
Shailesh
```

- For associative array, to loop oven an array, accessing both key and the value, use following syntax.
- **Syntax:**

```
foreach($array as $key => $var)
{
    //code to be executed
}
```

- **Alternative Syntax:**

```
foreach($array as $key => $var):
    //code to be executed
endforeach;
```

- The foreach loop also works with associative arrays, except that for such arrays, it uses two temporary variables - one each for the key and value.

- **Example:**

```
<?php
 $arr = array ('one'=>"Ritika", 'two'=> "Atharv", 'three'=>
"Shailesh");
 foreach ($arr as $key => $val) {
        echo "$key $val<br>";
    }
 ?>
```

- Here, foreach loop fetches one element from an array as key value pair into $key and $val variables respectively.
- **Note:** The foreach construct does not operate on the array itself, but rather on a copy of it.

Differences between for loop and foreach loop:

Table 2.2: Differences between for loop and foreach loop

Sr. No.	for Loop	foreach loop
1.	The for loop executes statement(s) repeatedly until specified expression evaluates to false.	The foreach loop executes a group of embedded statements for each element in an array.
2.	There is need to specify the loop bounds i.e. minimum and maximum.	There is no need to specify the loop bounds i.e. minimum and maximum.
3.	For this reason we use some counters in our loop which obviously occupies some memory.	We don't have any counter in this loop, so no extra memory required for the counters.
4.	The for loop needs initialization, condition checking as well as increment/decrement statement.	For for-each no need of initialization as well as increment/decrement statement.

5.	**Syntax:**
	for(initialization;condition;
	increment/decrement)
	{ statement(s); }
		Syntax:
		For Indexed array:
		foreach($arr as $val)
		{ statement(s); }
		
		For associative array:
		foreach($array as $key => $var){
		//code to be executed
		}

2.11.3 Iterator

- Arrays in PHP uses the pointer to keep track of the current element is known as the *iterator*. PHP has functions to set, move, and reset this iterator.
- Following are the iterator functions:
  - **current( ):** Returns the element currently pointed at by the iterator.
  - **reset( ):** Moves the iterator to the first element in the array and returns it.
  - **next( ):** Moves the iterator to the next element in the array and returns it.
  - **prev( ):** Moves the iterator to the previous element in the array and returns it.
  - **end( ):** Moves the iterator to the last element in the array and returns it.
  - **each( ):** Returns the key and value of the current element of an array and moves the iterator to the next element in the array.
  - **key( ):** Returns the key of the current element.
- The each( ) function is used to loop over the elements of an array. It processes elements according to their internal order.

Program 2.31: Write a PHP program to illustrate the use of iterator functions.

```
<?php
  $arr = array (5,10,15,20,25);
  echo "<br> Current element: ".current($arr);
  echo "<br> Next element: ".next($arr);
  echo "<br> Key of the Current element: ".key($arr);
  list($key,$value)=each($arr);
  echo "<br> Current element's Key and Value:$key $value ";
  echo "<br> Next element: ".next($arr);
  echo "<br> Previous element: ".prev($arr);
  echo "<br> Last element: ".end($arr);
  echo "<br> First element: ".reset($arr);
?>
```

Output:

```
Current element: 5
Next element: 10
Key of the Current element: 1
Current element's Key and Value:1 10
Next element: 20
Previous element: 15
Last element: 25
First element: 5
```

- Here in the beginning, current element will be the first element of the array.
- **Note:** This approach does not make a copy of the array, as foreach does. This is useful for very large arrays when you want to conserve memory.

2.11.4 Calling a function for each array elements

- **array_walk:** This function is used to call a user-defined function once per element in an array:
- **Syntax:** `array_walk(array, function_name);`
- For every element of the first argument *array*, *function_name* which is second argument will be invoked.
- The second argument function which will take two arguments: element's value and element's key.

Program 2.32: Write a PHP program to illustrate the use of `array_walk()` function.

```
<?php
  function display($val,$key)
  {
    echo "<br> $key => $val";
  }
  $arr = array ('One'=>"Ritika", 'Two'=> "Atharv", 'Three'=> "Shailesh");
  array_walk($arr,'display');
?>
```

Output:

```
One => Ritika
Two => Atharv
Three => Shailesh
```

2.11.5 Reduce an array

- **array_reduce:** A array_reduce( ) function is applied on each element of the array to build a single value.
- **Syntax:** `$var = array_reduce(array, function_name [, default ]);`
- The function takes two arguments: the running total, and the current value being processed. It returns the new running total.
- If third parameter is passed then it will be used as initial value for performing further calculations.

Program 2.33: Write a PHP program to illustrate the use of array_reduce() function.

```
<?php
function calculate ($running_total, $current_value) {
$running_total += $current_value * $current_value * $current_value;
return $running_total;
```

```
}
$a = array(1, 2, 3, 4, 5);
$result = array_reduce($a, 'calculate');
echo "<br> Addition of cube of all array elements: ".$result;
?>
```

Output:

Addition of cube of all array elements: 225

- Here, array_reduce( ) function makes following function calls:

  calculate(0,1)

  calculate(1,2)

  calculate (9,3)

  calculate (36,4)

  calculate(100,5)

  After the last call to calculate(100,5) will return 225 and will get assign to $result.
- Consider the array_reduce() function as follows:

  $result = array_reduce($a, 'calculate',50);

  Here 3rd argument i.e. default value is passed as argument then it makes following function calls:

  calculate(50,1)

  calculate(51,2)

  calculate (59,3)

  calculate (86,4)

  calculate(150,5)

  After the last call to calculate(150,5) will return 275 and will get assign to $result.
- If the array is empty, array_reduce( ) returns the default value. If no default value is given and the array is empty, array_reduce( ) returns NULL.

2.11.6 Search a value in an array

- **in_array():** This function checks the element is present in the array or not.
- **Syntax:** `Boolean_value = in_array(element, array [, strict]))`
- The first argument is the element to be searched into the array which is passed as second argument.
- This function returns true or false depending upon element is present in the array or not.
- If third argument is true then it will check the data type also.

Program 2.34: Write a PHP program to illustrate the use of in_array() function.

```
<?php
$arr = array (5,10,15,20,25);
$val=10;
if(in_array($val,$arr))
   echo "<br> $val is present in the array";
else echo "<br> $val is not present in the array";

$val="10";
if(in_array($val,$arr,true))        //checks value and data type
   echo "<br> String $val is present in the array";
else echo "<br> String $val is not present in the array";

$val=13;
if(in_array($val,$arr))
   echo "<br> $val is present in the array";
else echo "<br> $val is not present in the array";
?>
```

Output:

10 is present in the array

String 10 is not present in the array

13 is not present in the array

- **array_search():** This function checks the element is present in the array or not like `in_array()` function, but this function return index of the element if it is present.
- **Syntax:** `index = array_search(array, element, [, strict]))`
- The first argument is an array in which the element which is passed as second argument to be searched.
- If third argument is true then it will check the data type also.

Program 2.35: Write a PHP program to illustrate the use of in_array() function.

```
<?php
$arr = array (5,10,15,20,25);
$val=20;
$key=array_search($val,$arr);
if($key)
```

```
    echo "<br> $val is present in the array at index $key";
else echo "<br> $val is not present in the array";

$val="20";
$key=array_search($val,$arr,true);  //checks value and data type
if($key)
    echo "<br> string $val is present in the array at index $key";
else echo "<br> String $val is not present in the array";
?>
```

Output:

20 is present in the array at index 3

String 20 is not present in the array

2.12 SORTING USING ARRAYS

- In PHP, there are different ways to sort given data in ascending or descending order. Sorting can be based on keys or values. Depending upon need, keys can be kept unchanged after sorting.
- Sorting functions can be used to sort strings in alphabetical order, numbers from smallest to biggest or biggest to smallest etc.

2.12.1 Sorting One array at a Time

- There are three set of functions exists to sort the given data:
  - sorting by values and then changing the keys.
  - sorting by values without changing the keys.
  - sorting by keys.

Table 2.3: Sorting Functions

Purpose	Function used to sort in Ascending order	Function used to sort in Descending order	Function used to sort in order defined by user
sorting by values and then changing the keys	sort( )	rsort( )	usort( )
sorting by values without changing the keys	asort( )	arsort( )	uasort( )
sorting by keys	ksort( )	krsort( )	uksort( )

- The sort(), rsort() and usort() functions are used to sort indexed array in ascending order, descending and user defined sorting order respectively.

Program 2.36: Write a PHP program to illustrate sorting of an indexed array in ascending order, descending order and used defined order.

```
<?php
$age=array(20,15,40,37);
echo "<br> Original Array<br>";
print_r($age);

echo "<br> Array sorted in Ascending order<br>";
sort($age);
print_r($age);

echo "<br> Array sorted in Descending order<br>";
rsort($age);
print_r($age);

$c=array("Pune","Nagpur", "Mumbai","Nashik");
echo "<br><br> Original Array<br>";
print_r($c);

echo "<br> Array sorted in Ascending order<br>";
sort($c);
print_r($c);

echo "<br> Array sorted in Descending order<br>";
rsort($c);
print_r($c);

function usersort($a,$b)
{
  if($a==$b) return 0;
   return ($a>$b)?-1:1;
}
echo "<br><br> Array sorted in descending order using user defined
function<br>";
usort($age,"usersort");
print_r($age);
?>
```

Output:

Original Array
Array ([0] => 20 [1] => 15 [2] => 40 [3] => 37)
Array sorted in Ascending order
Array ([0] => 15 [1] => 20 [2] => 37 [3] => 40)
Array sorted in Descending order
Array ([0] => 40 [1] => 37 [2] => 20 [3] => 15)

Original Array
Array ([0] => Pune [1] => Nagpur [2] => Mumbai [3] => Nashik)
Array sorted in Ascending order
Array ([0] => Mumbai [1] => Nagpur [2] => Nashik [3] => Pune)
Array sorted in Descending order
Array ([0] => Pune [1] => Nashik [2] => Nagpur [3] => Mumbai)

Array sorted in descending order using user defined function
Array ([0] => 40 [1] => 37 [2] => 20 [3] => 15)

- The asort() and arsort() functions are used to sort associative array in ascending and descending order of values respectively. Here keys are preserved, they do not change.
- Similarly ksort() and krsort() functions are used to sort associative array in ascending and descending order of keys respectively. Here keys are preserved, they do not change.

Program 2.37: Write a PHP program to illustrate sorting of an associative array in ascending order and descending order based on values and keys.

```
<?php
$stud=array("Atharv"=>"SYBCA", "Shailesh"=>"TYBCA","Ritika"=>"FYBCA",);
echo "<br><br> Original Array<br>";
print_r($stud);

echo "<br><br> Array sorted in Ascending order of values<br>";
asort($stud);
print_r($stud);

echo "<br><br> Array sorted in Descending order of values<br>";
arsort($stud);
print_r($stud);

echo "<br><br> Array sorted in Ascending order of keys<br>";
ksort($stud);
print_r($stud);

echo "<br><br> Array sorted in Descending order of keys<br>";
krsort($stud);
print_r($stud);
?>
```

Output:

Original Array
Array ([Atharv] => SYBCA [Shailesh] => TYBCA [Ritika] => FYBCA)
Array sorted in Ascending order of values
Array ([Ritika] => FYBCA [Atharv] => SYBCA [Shailesh] => TYBCA)

Array sorted in Descending order of values
Array ([Shailesh] => TYBCA [Atharv] => SYBCA [Ritika] => FYBCA)

Array sorted in Ascending order of keys
Array ([Atharv] => SYBCA [Ritika] => FYBCA [Shailesh] => TYBCA)

Array sorted in Descending order of keys
Array ([Shailesh] => TYBCA [Ritika] => FYBCA [Atharv] => SYBCA)

2.12.2 Natural Order Sorting

- **natsort():**When strings are alphanumeric then built-in sorting functions doesn't work correctly. Such strings are sorted using `natsort()` function.
- **Syntax:** `natsort(array);`

  `nastcasesort(array);`
- The function `natcasesort()` can be used to ignore the case of strings.

Program 2.38: Write a PHP program to illustrate sorting of an alphanumeric strings.

```
<?php
$username=array("user13","user9","user7","user10");

echo "<br> Array of User Names<br>";
print_r($username);

echo "<br><br> Array of User Names using sort()<br>";
sort($username);
print_r($username);

natsort($username);
echo "<br><br> Array after sorting <br>";
print_r($username);
?>
```

Output:

Array of User Names
Array ([0] => user13 [1] => user9 [2] => user7 [3] => user10)

Array of User Names using sort()
Array ([0] => user10 [1] => user13 [2] => user7 [3] => user9)

Array after sorting
Array ([2] => user7 [3] => user9 [0] => user10 [1] => user13)

2.12.3 Sorting Multiple Arrays at Once

- **array_multisort( ):** This function sorts multiple indexed arrays at once.
- **Syntax:** `array_multisort(array1 [, array2, ... ]);`
- Pass different arrays as parameters and sorting order which can be SORT_ASC or SORT_DESC constants.
- The first element of each array represents a single record. Similarly, the second element constitutes another record and so on.
- This function reorders the elements of the arrays, preserving the records and assigns new indices.

Program 2.39: Write a PHP program to illustrate the `array_multisort()` function.

```
<?php
$name=array("Atharv","Shailesh","Ritika","Pravin");
$class=array("SYBCA","TYBCA","FYBCA","FYBCA");
$age=array(20,21,19,18);
echo "<br> Array of Name<br>";
print_r($name);
echo "<br> Array of Class<br>";
print_r($class);
echo "<br> Array of Age<br>";
print_r($age);

array_multisort($class, SORT_ASC, $age, SORT_ASC,$name,SORT_DESC);

echo "<br><br>Arrays after Multisort()";
echo "<br> Array of Name<br>";
print_r($name);
```

```
echo "<br> Array of Class<br>";
print_r($class);
echo "<br> Array of Age<br>";
print_r($age);
?>
```

Output:

```
Array of Name
Array ( [0] => Atharv [1] => Shailesh [2] => Ritika [3] => Pravin )
Array of Class
Array ( [0] => SYBCA [1] => TYBCA [2] => FYBCA [3] => FYBCA )
Array of Age
Array ( [0] => 20 [1] => 21 [2] => 19 [3] => 18 )

Arrays after Multisort()
Array of Name
Array ( [0] => Pravin [1] => Ritika [2] => Atharv [3] => Shailesh )
Array of Class
Array ( [0] => FYBCA [1] => FYBCA [2] => SYBCA [3] => TYBCA )
Array of Age
Array ( [0] => 18 [1] => 19 [2] => 20 [3] => 21 )
```

2.12.4 Reversing an Array

- **array_reverse( ) :** The array_reverse( ) function reverses the internal order of elements in an array. Numeric keys are renumbered starting at 0, while string indexes are unaffected. Return value will be the reversed array.
- **Syntax:** `array array_reverse(array);`

Program 2.40: Write a PHP program to illustrate the use of array_reverse( ) function.

```
<?php
$a1=array(2,3,4,7,9);
echo "<BR> Array a1: <BR>";
print_r($a1);

$a1rev=array_reverse($a1);
echo "<BR> Array a1 after Reverse: <BR>";
print_r($a1rev);

$a2=array("a"=>"String1","b"=>"String2","c"=>"String3","d"=>"String4");
```

```
echo "<BR> Array a2: <BR>";
print_r($a2);

$a2rev=array_reverse($a2);
echo "<BR> Array a2 after Reverse: <BR>";
print_r($a2rev);
?>
```

Output:

Array a1:
Array ([0] => 2 [1] => 3 [2] => 4 [3] => 7 [4] => 9)
Array a1 after Reverse:
Array ([0] => 9 [1] => 7 [2] => 4 [3] => 3 [4] => 2)
Array a2:
Array ([a] => String1 [b] => String2 [c] => String3 [d] => String4)
Array a2 after Reverse:
Array ([d] => String4 [c] => String3 [b] => String2 [a] => String1)

- **array_flip( ) :** The `array_flip( )` function returns an array that reverses the order of each element's key-value pair:
- **Syntax:** `array = array_flip(array);`
- It takes an array as argument and returns an array whose keys and values are interchanged i.e. for each element of an array whose value is a valid key, the element's value becomes its key and the element's key becomes its value.
- The `array_reverse( )` function reverses the internal order of elements in an array. Numeric keys are renumbered starting at 0, while string indexes are unaffected.

Program 2.41: Write a PHP program to illustrate the use of array_flip( ) function.

```
<?php
$a1=array(10,20,30,40,50);
echo "<BR> Array 1: <BR>";
print_r($a1);

$a1rev=array_flip($a1);
echo "<BR> Array 1 after array_flip: <BR>";
print_r($a1rev);

$a2=array("a"=>"String1","b"=>"String2","c"=>"String3","d"=>"String4");
echo "<BR><BR> Array 2: <BR>";
print_r($a2);

$a2rev=array_flip($a2);
echo "<BR> Array 2 after array_flip: <BR>";
print_r($a2rev);
?>
```

Output:

Array 1:
Array ([0] => 10 [1] => 20 [2] => 30 [3] => 40 [4] => 50)
Array 1 after array_flip:
Array ([10] => 0 [20] => 1 [30] => 2 [40] => 3 [50] => 4)

Array 2:
Array ([a] => String1 [b] => String2 [c] => String3 [d] => String4)
Array 2 after array_flip:
Array ([String1] => a [String2] => b [String3] => c [String4] => d)

2.12.5 Randomizing Order

- **shuffle( ):** To traverse the elements in an array in a random order, use the shuffle( ) function. All existing keys, whether string or numeric, are replaced with consecutive integers starting at 0.
- **Syntax:** `shuffle(array)`
- The array which is passes as parameter is modified.

Program 2.42: Write a PHP program to illustrate the use of shuffle( ) function.

```
<?php
$a1=array(2,3,4,7,9);
echo "<BR> Array a1: <BR>";
print_r($a1);

shuffle($a1);
echo "<BR> Array a1 after Shuffle: <BR>";
print_r($a1);

$a2=array("a"=>"String1","b"=>"String2","c"=>"String3","d"=>"String4");
echo "<BR> Array a2: <BR>";
print_r($a2);

shuffle($a2);
echo "<BR> Array a2 after Shuffle: <BR>";
print_r($a2);

?>
```

Output:

Array a1:
Array ([0] => 2 [1] => 3 [2] => 4 [3] => 7 [4] => 9)
Array a1 after Shuffle:
Array ([0] => 9 [1] => 7 [2] => 2 [3] => 3 [4] => 4)
Array a2:
Array ([a] => String1 [b] => String2 [c] => String3 [d] => String4)
Array a2 after Shuffle:
Array ([0] => String1 [1] => String2 [2] => String4 [3] => String3)

2.12.6 Recursive Function

- A recursive function is a function that calls itself repeatedly until a condition is satisfied. Such a function is typically used to solve complex and iterative calculations.

Program 2.43: Write a program of factorial number using recursive function.

```
<?php
function factorial($n)
{
    if ($n == 0|| $n==1)
        return 1;                          // Terminating condition
    return ($n * factorial ($n -1));  //  Recursive call
}

$no=4;
if($no < 0 )
        echo "Invalid Input";
else
        echo "Factorial of $no is ".factorial($no);
?>
```

Output:

Factorial of 4 is 24

PROGRAMS

Program 2.44: Write a PHP script to find the factorial of a number.

```
<?php
function fact($no)
{
    $result=1;
    if($no==0 || $no==1)
        return $result;
```

```
    else
    {
        for($i=2;$i<=$no;$i++)
          $result=$result* $i;
        return $result;
    }
}
$no=5;
if($no<0)
   echo "Factorial can't be find out";
else
{
   $result=fact($no);
   echo "<BR> Factorial of $no is $result ";
}
?>
```

Output: Factorial of 5 is 120

Program 2.45: Write a PHP script to find the reverse of a number.

```
function reverse($no)
{
   $temp=$no;
   $revno=0;
   while($no>0)
   {
      $rem=$no%10;
      $revno=$revno*10+$rem;
      echo "<BR> $revno $no";
      $no=(int)($no/10);
   }
   return $revno;
}
   $no=123;
   $result=reverse($no);
   echo "<BR> Reverse of $no is $result ";
?>
```

Output: Reverse of 123 is 321

Program 2.46: Write a PHP script to check whether a number is prime or not.

```
<?php
function IsPrime($no)
{
  $mid=$no/2;
  for($i=2;$i<=$mid;$i++)
  {
     if($no%$i==0)
          return 0;
  }
   return 1;
}
$no=5;
if(IsPrime($no))
    echo "$no is prime";
else echo "$no is not prime";
   echo "<BR>";
   $no=9;
   if(IsPrime($no))
       echo "$no is prime";
   else echo "$no is not prime";
?>
```

Output:

5 is prime

9 is not prime

Program 2.47: Write a PHP script to check whether a string is palindrome or not.

```
<?php
function checkpalindrome($str)
{
   $len=strlen($str);
   //without built-in function
   for($i=0,$j=$len-1;$i<$len/2 && $j>=0;$i++,$j--)
   {
       if($str[$i]!=$str[$j])
       {
          echo "String $str is not  palindrome ";
          return;
       }
```

```
    }
    echo "String $str is Palindrome ";
}
function funpalindrome($str)
{
    if($str!=strrev($str))          // using built in function
    {
      echo "String $str is not  palindrome ";
      return;
    }
    echo "String $str is Palindrome ";
  }

 $str="Hello";
  checkpalindrome($str);
  echo "<BR>";

 $str="madam";
  checkpalindrome($str);
?>
```

Output:

String Hello is not palindrome

String madam is Palindrome

Summary

- Functions in PHP can be defined with keyword function. Parameters can be passed to a function in two ways –pass by value and pass by reference. Also a function can return a single value, multiple values and even reference as well.
- A function can have default arguments. In a function call some parameters can be missing that is PHP allows to pass less number of arguments than what is written in the function definition.
- PHP supports different functions to store data in array like range() and arra_pad(). Also there are built-in functions sizeof() and count() to get number of elements of an array.
- Built-in functions like array_slice(), array_chunk(), array_splice(), array_keys(), array_values(), in_array(), array_search() are also supported by PHP.
- Also we have studied array_reduce(), array_walk(), array_reverse(), shuffle() and different functions to sort data by values and by keys.

Trace the Output

1)
```
<?php
$a1=array(1,2,3,3,5);
$slice1 = array_slice($a1, 3);
print_r($slice1);
?>
```

Output: Array ([0] => 3 [1] => 5)

2)
```
<? Php
function makeice ($ flavour, $type = "choco")
{
return "making a bowl of $type $ flavour \n",
}
echo makeice ("raspbery") ;
?>
```

Output: making a bowl of choco raspbery

3)
```
<?php
$str1="Hi";
function demo($str2)
{
    echo $str2;
    echo $str1;
}
demo("Hello");
?>
```

Output: Hello

4)
```
<?php
$a=range(0,5);
print_r($a);
?>
```

Output: Array([0]=> 0[1]=>1[2]=>2[3]=>3[4]=>4[5]=>5)

5)
```
<?php
$alphabet = array ("A", "B", "C");
echo (next($alphabet));
?>
```

Output: B

6)
```
<?php
$fruits = array ("mango", "apple", "pear", "peach");
$fruits = array_flip($fruits);
echo ($fruits["apple"]);
?>
```

Output: 1

Check Your Understanding

Q.I Multiple choice questions

1. Basic syntax for calling a function is _____.
   (a) function_name (experssion1, expersion2)
   (b) (experssion1 : Function_name, expersion2 : Function_name)
   (c) (experssion1, expersion2)
   (d) None of them
2. Which function will return true if a variable is an array or false otherwise?
   (a) this_array( )
   (b) is_array( )
   (c) unset( )
   (d) in_array( )
3. Which of the following PHP function is used to build a function that accepts any number of arguments?
   (a) func_get_argv( )
   (b) func_get_args( )
   (c) get_argv( )
   (d) get_argc( )
4. By default, the index of array in PHP starts from _____.
   (a) 0
   (b) 1
   (c) –1
   (d) 2

5. Which of the following are correct ways of creating an array?
   (i) arr[0] = "letsfindcourse";
   (ii) $arr[] = array("letsfindcourse");
   (iii) $arr[0] = "letsfindcourse";
   (iv) $arr = array("letsfindcourse");
   (a) (ii), (iii) and (iv)
   (b) (ii) and (iii)
   (c) Only (i)
   (d) (iii) and (iv)

ANSWERS

1. (a)	2. (b)	3. (b)	4. (a)	5. (d)

Q.II State True/False.

1. "A function can have variable number of arguments".
2. "Elements can't be added in an array without using index value".
3. The function call range(2,5) will create an array : (5,4,3,2).
4. The array_chunk() will return subarrays with equal number of elements, last chunk can have less number of elements
5. The array_splic() function is used to insert as well as to remove elements from an array.

ANSWERS

1. True	2. False	3. False	4. True	5. True

Practice Questions

Q.I Answer the following questions in short.

1. Differentiate between indexed array and associative array.
2. How to find the number of parameters passed to the function?
3. What is the use of array_slice( ) function?
4. What is an anonymous function?
5. Differentiate between anonymous function and normal function.
6. Explain working of list() function.

7. How to get array values and keys into separate arrays?
8. Compare for loop and foreach loop.
9. What is the difference between in_array() and array_search() function?
10. Explain array_flip() function with syntax.

Q.II Answer the following questions.

1. Explain the concept of missing parameters to a function with suitable example.
2. Explain with suitable example.
   `range( ), array_slice( ), array_reverse( ), array_flip(), shuffle( )`
3. Write the syntax and purpose of the following functions along with example.
   `arr_pad(), sizeof(),count(), array_chunk(), array_splice(), array_keys(), array_values()`
4. Write the syntax and purpose of the following functions along with example.
   `in_array(), array_search(), array_reduce(), array_walk(), array_reverse()`
5. Explain different sorting functions with example.
6. Define an array. Find the elements from the array that matches the given value using appropriate search function.
7. Write a menu driven program to perform the following operations on associative arrays:
   (a) Sort the array by values (changing the keys) in ascending, descending order.
   (b) Also sort the array by values without changing the keys.
   (c) Sort the different arrays at a glance using single function.
8. Explain different functions to extract values from an array.
9. Explain different iterator functions in PHP with example.
10. What is multidimensional array? How to access elements from multidimensional array?
11. Write a PHP script with user defined function to find the addition of digits of a number.
12. Write a PHP script to sort array of marks. (Array contains names and marks).
13. Which function is used to insert as well as remove elements from an array? Explain with example.

14. Write a menu driven program to perform the following operations on an associative array:
    (a) Display the elements of an array along with the keys.
    (b) Display the size of an array
    (d) Reverse the order of each element's key-value pair.
    (e) Traverse the elements in an array in random order.

Q.III Define the terms

1. Array
2. Associative array
3. Indexed array
4. Anonymous function
5. Default parameters
6. Multidimentional array
7. Iterator
8. Natural order Sorting

***

3... Object Oriented Programming

Objectives...

- To learn the benefits of encapsulating code into functions and classes.
- To define and use user-defined classes with properties and methods.
- To understand OOP concepts of visibility, inheritance and interface.
- To study built-in functions for examining classes' and object's characteristics.
- To learn the concept of byte stream representation of an object using serialization.

3.1 INTRODUCTION

- Like C++ and Java, PHP also supports Object Oriented Programming (OOP) concepts.
- OOP was introduced in PHP3 and improved further. OOP supports cleaner designs, easier maintenance and greater code reuse. It also speeds up the development of large applications.
- It includes the fundamental connection between data and the code that works on that data. The class and object are the fundamental construct behind object oriented programming.
- Objects can represent real world entity like student, employee or it can represent conceptual entity link bank account, file. Each object has different values for properties it possesses.
- Objects with similar characteristics are grouped together into a single unit called **Class**. The class can contain methods which represents the way in which the object can interact with its data.
- **Advantages of Object-Oriented Programming**:
  1. **Modularization:** The application can be divided into modules.
  2. **Re-usability:** An application can be implemented by adding new modules to the existing modules. This speeds up the application development time.

3.2 CLASSES

- PHP allows you to group together objects having similar characteristics and behaviour into a single unit called as *class*.
- Class is a programmer defined data type, which includes variables and functions.
- Variables within a class are called *properties* which have a name and a value; functions are called *methods* which represent the actions associated with the class. Methods can accept parameters of any valid data type.
- Class is a collection of objects. An object has properties and behaviour.
- Class names are case insensitive and must conform to the rules for PHP identifiers.
- We define our own class by starting with the keyword **'class'** followed by the name class name. Inside a class, you can define variables and functions as shown below:
- **Syntax:**

```
<?php
class class_name
{
    // Properties
    // Methods
}  //end of class
?>
```

3.2.1 Declaring Properties

- Property of a class can be defined as private, public or protected. Property declaration is optional. It's a good programming style to declare it. You can add new properties at any time.
- You can assign default values to properties, but those default values must be simple constants.
- **Example:**

```
public $name= "Ritika";
public $age = 18;
```

- Using access modifiers, you can change the visibility of properties.
- Properties that are accessible outside the class using object should be declared as *public*; properties on an instance that can only be accessed by methods within the same class should be declared *private*. Finally, properties declared as *protected* can only be accessed by the object's class methods and the class methods of the classes inheriting from the class.
- Defining the visibility of class properties is optional; if a visibility is not specified, it is *public* by default.

- **Example:**

```
class Demo
{
    private $name = "Atharv";
    protected $age = 20;
    public $class = "SYBBA(CA)";
}
```

- Here $name is accessible inside class methods only. $age is accessible inside class methods and class methods of classes inheriting from class Demo. $class is accessible outside the class using an object of the class type.

3.2.2 Declaring Methods

- A method is a function defined inside a class. In PHP, most methods act only on data of a class in which the method resides.
- Within a method, the $this variable contains a reference to an object which invokes the method. Using $this, variables of an object can be accessed inside the member function of a class.
- Using access modifiers, you can change the visibility of methods. Methods that are accessible outside the class using object should be declared as *public*; methods on an instance that can only be called by methods within the same class should be declared *private*. Finally, methods declared as *protected* can only be called from within the object's class methods and the class methods of classes inheriting from the class.
- Defining the visibility of class methods is optional; if a visibility is not specified, a method is *public*.
- For Example: PHP script to define a class Employee. With methods to accept and display employee details.

```
<?php
class Employee
{
private $emp_code, $emp_name, $emp_designation;  // property declaration
public function accept($c,$n,$d)                  //method definition
{
    $this->emp_code=$c;  //$this is used to access properties of
                                 current invoking object

    $this->emp_name=$n;
    $this->emp_designation=$d;
}
   function display()   //default visibility is public
   {
```

```
        echo "<br>Emp Code: ".$this->emp_code;
        echo "<br>Emp Name: ".$this->emp_name;
        echo "<br>Emp Designation: ".$this->emp_designation;
    }
}
```

- If you call `$emp->accept()` then inside accept method, $this hold the same value as $emp. Methods use the `$this` variable to access the properties of the current object. As you can see, the `accept()` and `display()` methods use `$this` to access and set `emp_code`, `emp_name` and `emp_designation` properties of the current object.

3.2.3 Static Property and Static Method

- PHP allows you to define static properties, which are variables of class and can be accessed by referencing the property with the class name.
- Also to access methods in terms of a class rather than an object, use **static** keyword. Any method declared as static is accessible without the creation of an object. Static functions are associated with the class, and not with an instance of the class. They are permitted to access only static methods and static variables.
- Inside a class, you can refer to the static property using the **self** keyword.
- **Syntax**: `self::static_var_name;`

Program 3.1: Write a PHP script to illustrate the concept of static property and static method.

```
<?php
class example_static
{
    static $count;   //static property
    public static function updateCount()
    {
        //static method
        return self::$count++;
    }
}
example_static::$count = 1;
for($i = 0; $i < 5; ++$i)
{
    echo 'The value is: '.example_static::updateCount() . "<BR>";
}
?>
```

Output:

The value is: 1
The value is: 2
The value is: 3
The value is: 4
The value is: 5

3.3 OBJECTS

- An individual instance of the data structure defined by a class is called as Object. You define a class once and then declare objects of that class. Building an object using a class is known as Instantiation.
- Instantiating an object requires:
  - Memory allocation into which load the object.
  - The data that will populate the values of the properties.
- An object has its own individual identity. Each object contains data and code to manipulate the data.
- Objects of different classes can interact with each other without knowing details of other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Declaring an Object:

- To declare an object of a given class, the `new` keyword is used:
- **Syntax:**

```
$object = new Class;
```

- **Example:**

```
$emp = new Employee;
```

Accessing Properties and Methods:

- Class methods and properties can be directly accessed through the object instance using the object access operator (->) notation.
- **Syntax:**

```
$object->property_name      //if visibility of property is public
$object->method_name([arg, ... ])
```

- **Example:**

```
$emp->accept(1,"Atharv", "Manager");
$emp->display();
```

- **Note:** Classes are manipulated at design time when you make changes to the methods or properties and objects are manipulated at runtime when values are assigned to their properties and their methods are invoked.

Initializing an object using Constructor:

- To initialize the object, Constructor is special type of function of a class which is automatically executed as any object of that class is created or instantiated. Usually it starts with two underscore characters.
- **Syntax:**

```
function __construct(args..)
{
    //definition
}
```

- You may provide a list of arguments following the class name when instantiating an object. These arguments are passed to a constructor that initializes the properties of the class.

Destroying an object using Destructor:

- When an object is destroyed, such as when the requested page has completed running, when the variable falls out of scope, when it is explicitly set to null, or the end of the script is reached, its destructor is called. Destructor does not take any parameters.
- **Syntax:**

```
function __destruct()
{
    //definition
}
```

- This helps to perform any last minute clean up, such as closing file handle or database connections that might have been opened using class.

Program 3.2: Write a PHP script to define a class Employee with `emp_code`, `emp_name` and `emp_designation` as data members. Define constructor and destructor for the class. Also define display() function to display employee details.

```
<?php
class Employee
{
    private $emp_code;
    private $emp_name;
    private $emp_designation;
    function __construct($c,$n,$d)     //constructor definition
    {
        $this->emp_code=$c;
        $this->emp_name=$n;
        $this->emp_designation=$d;
        echo "<br> Object is created";
    }
    function display()
    {
        echo "<br>Emp Code: ".$this->emp_code;
        echo "<br>Emp Name: ".$this->emp_name;
        echo "<br>Emp Designation: ".$this->emp_designation;
    }
```

```
    function __destruct()   //destructor definition
    {
        echo "<br>Object is destroyed";
    }
}
$c=1;
$n="Atharv ";
$d="Manager";
//constructor called automatically when object is created
$emp=new Employee($c,$n,$d);
$emp->display();
?>
```

Output:

Object is created
Emp Code: 1
Emp Name: Atharv
Emp Designation: Manager
Object is destroyed

In above PHP script, the statement `$emp=new Employee($c,$n,$d)` invokes the constructor function automatically when object is created and at the end of the script destructor is called automatically.

Program 3.3: Write a PHP script to define a class Rectangle with length and breadth as data members. Define constructor and destructor for the class. Also define `area()` and `perimeter()` functions to display area and perimeter of a rectangle.

```
<?php
class Rectangle
{
  function Rectangle($l,$b)
  {
    $this->l=$l;
    $this->b=$b;
    echo "<br> Object is initialized";
    echo "<br> Length = $this->l, Breadth= $this->b";
  }
  function area()
  {
    $a=$this->l*$this->b;
    echo "<br>Area of rectangle =$a";
```

```
    }
    function perimeter()
    {
        $p=2*($this->l+$this->b);
        echo "<br>Perimeter of rectangle =$p";
    }
    function __destruct()   //destructor definition
    {
        echo "<br>Object is destroyed";
    }
}
$l=3.2;
$b=5.4;
$r=new Rectangle($l,$b);
$r->area();
$r->perimeter();
?>
```

Output:

Object is initialized

Length = 3.2, Breadth = 5.4

Area of rectangle = 17.28

Perimeter of rectangle = 17.2

Object is destroyed.

In above PHP script, the statement `$r=new Rectangle($l,$b)` invokes the constructor function automatically when object is created. The object `$r` of a class Rectangle invokes `area()` and `perimeter()` method using -> symbol and displays the output. At the end of the script destructor is called automatically.

3.4 INTROSPECTION

- It is the ability of a program to examine an object's characteristics, such as its name, parent class (if any), properties and methods.
- With introspection, you can write code that operates on any class or object. You don't need to know which methods or properties are defined when you write your code; instead, you can discover that information at runtime.

Examining Classes:

- To determine whether a class exists, use the `class_exists()` function, which takes in a string and returns a Boolean value.

  Syntax: `$yes_no = class_exists(classname);`

- Alternately, you can use the `get_declared_classes()` function, which returns an array of defined classes and you can verify the class name in the returned array:
  Syntax: `$classes = get_declared_classes();`
- You can get the methods and properties that exist in a class (including those that are inherited from super classes) using the `get_class_methods()` and `get_class_vars()` functions. These functions take a class name and returns an array:
  Syntax: `$methods = get_class_methods(classname);`
  `$properties = get_class_vars(classname);`
- **Note:** `get_class_vars()` returns only properties that have default values; there's no way to discover uninitialized properties.
- Use `get_parent_class()` to find a class's parent class name.
  Syntax: `$superclass = get_parent_class(classname);`

Examining an Object:

- To get the class to which an object belongs, first check if it is an object using the `is_object()` function, then get the class with the `get_class()` function:
  Syntax: `$yes_no = is_object(var);`
  `$classname = get_class(object);`
- Before calling a method on an object, you can check that it exists using the `method_exists()` function:
  Syntax: `$yes_no = method_exists(object, method);`
  Calling an undefined method triggers a runtime exception.
- `get_object_vars()` returns an array of properties that are set in an object.
  Syntax: `$array = get_object_vars(object);`

Program 3.4: Write a PHP script to illustrate the use of functions to examine characteristics of classes and objects.

```
<?php
class A
{
  private $x=10, $y;
  function A($a)
  {
      $this->y=$a;
      echo "<BR> Constructor A is called";
  }
  function displayA()
  {
      echo "<BR> x=$this->x y=$this->y";
  }
}
class B extends A
```

```
{
    private $p,$q;
    function B($l,$m,$n)
    {
        parent::A($l);
        $this->p=$m;
        $this->q=$n;
        echo "<BR> Constructor B is called";
    }
    function displayB()
    {
        echo "<BR> p=$this->p q=$this->q";
    }
}
$obj = new B(20,30,40);
$obj->displayA();
$obj->displayB();
echo "<BR>Functions examining classes<BR>";
echo "<BR> Class A exists or not: ";
echo class_exists('A');
echo "<BR> Class C exists or not: ";
echo class_exists('C');
echo "<BR> Get existing all classes:<BR>";
$allclasses= get_declared_classes();
echo "<BR> Get methods of class A:<BR>";
print_r(get_class_methods('A'));
echo "<BR> Get methods of class B:<BR>";
print_r(get_class_methods('B'));

echo "<BR> Get variables of class A<BR>";
var_dump(get_class_vars('A'));
echo "<BR> Get variables of class B(no property with default value)<BR>";
print_r(get_class_vars('B'));
echo "<BR> Parent of class B";
echo "<br>".get_parent_class($obj);  //from object of class B
echo "<br>".get_parent_class('B');   // from class name
echo "<BR> Parent of class A";  //no parent so no output
echo get_parent_class('A');
```

```
//Functions examining objects
echo "<BR>obj is object or not: ";
echo is_object($obj);
echo "<BR>obj1 is object or not: ";  //obj1 is not declared an object
echo is_object($obj1);
echo "<BR>Get obj's class name: ";
echo get_class($obj);
echo "<BR>Method displayA() exists or not: ";
echo method_exists($obj, 'displayA');
echo "<BR>Method displayC() exists or not: ";
echo method_exists($obj, 'displayC');
echo "<BR> Get variables of class B using object(no property with default
value) :<BR>";
print_r(get_object_vars($obj));
?>
```

Output:

Constructor A is called

Constructor B is called

x=10 y=20

p=30 q=40

Functions examining classes

Class A exists or not: 1 // Class A exists

Class C exists or not: // Class C does not exists

Get existing all classes:

Get methods of class A:

Array ([0] => A [1] => displayA) // Methods of a class A are returned as array elements

Get methods of class B:

Array ([0] => B [1] => displayB [2] => A [3] => displayA)

// Methods of a class B are its own methods along with methods of base class A are returned as array elements

Get variables of class A

array(0) { } //plz check it should output array([x] => 10)

Get variables of class B(no property with default value)

Array( ) //plz check it should output array([x] => 10)
Parent of class B
A // from object of class B
A // from class name B
Parent of class A: //As class A doesn't have base class, no output is displayed
obj is object or not: 1 //obj is declared so output is 1
obj1 is object or not: //obj1 is not declared so error is displayed

Get obj's class name: B
Method displayA( ) exists or not: 1 //displayA() method exists so output is 1
Method displayC( ) exists or not: //displayC() method doesn't exists so no output

Get variables of class B using object(no property with default value) :
Array () //plz check it should output array([x] => 10)

3.5 SERIALIZATION

- The conversion of any PHP value to a byte stream representation is called as Serialization. This byte stream can be stored in a file.
- **Syntax:**

```
string serialize(value);
```

- When an object is serialized, it will save all variables in an object. Also, it stores the name of the class. The methods in an object will not be saved.
- To convert a byte stream representation back to a PHP value, use `unserialize()` function.
- When you unserialize an object, its class must be defined before unserialization occurs. Unserializing an object whose class is not yet defined puts the object into `stdClass`, which is useless.
- **Syntax:**

```
value unserialize(string);
```

Program 3.5: Write a PHP script to illustrate the use of serialize() and unserialize() functions.

```
<?php
class A
{
 private $x=10, $y;
 function A($a)
 {
     $this->y=$a;
     echo "<BR> Object is Initialized";
 }
 function displayA()
```

```
    {
        echo "<BR> x=$this->x y=$this->y";
    }
}
$obj=new A(20);
echo "<BR> Object before writing to a file";
$obj->displayA();
$bytestream=serialize($obj);   //$obj converted to a bytestream
echo "<BR><BR> Bytestream representation: $bytestream";
$fp=fopen("test.txt","w");
fwrite($fp,$bytestream);        //bytestream is written to a file
fclose($fp);
$fp=fopen("test.txt","r");
$bytestream1=fread($fp,filesize("test.txt")); //read bytestream from a file
fclose($fp);
echo "<BR>";
$obj1=unserialize($bytestream1);  //converts bytestream back to an object
echo "<BR> Object after reading from a file";
$obj1->displayA();
echo "<BR><BR>Object using var_dump:<BR> ";
var_dump($obj1);                  // displays details of $obj1
?>
```

Output:

```
Object is Initialized
Object before writing to a file
x=10 y=20

Bytestream representation: O:1:"A":2:{s:4:"Ax";i:10;s:4:"Ay";i:20;}
Object after reading from a file
x=10 y=20
Object using var_dump:
object(A)#2 (2) { ["x":"A":private]=> int(10) ["y":"A":private]=> int(20) }
```

- In previous PHP script, `$obj` is converted to a byte stream using `serialize()` function. Then this `byte stream` can be used anywhere. In this case is stored in a file. Read this byte stream stored in a file and use `unserialize()` function to convert it back to an object. Then any member functions can be invoked using this object.

3.6 INHERITANCE

- Inheritance is an important feature of Object-Oriented Programming.
- The capability of a class to derive properties and methods from another class is called Inheritance.
- The class whose properties and method are inherited into a new class is called as base class and the class which inherits is called as derived class.
- The derived class does not have to define all the properties and functions again and again, as these are inherited from base class that possesses it. This allows re-usability of the code and speeds up the overall development process.
- To inherit the properties and methods from one class into another class, use **extends** keyword in the class definition, followed by the name of the base class.
- The public property or method of base class will be public in the derived class. The protected members will remain protected after inheritance. The private property will not be inherited, but it can be manipulated through public inherited methods of base class. The private methods of base class can be invoked by methods of derived class.
- **Syntax:**

```
class derived_classname extends base_classname
{
    //properties
    //methods
}
```

- **Example:**

```
class Employee
{
    private $emp_code, $emp_name, $emp_designation;
}
class EmpoyeeSalary extends Employee
{
    private $basic_pay, $earning, $deduction;
}
```

- The `classEmpoyeeSalary` contains the `$basic_pay`, `$earning` and `$deduction`, as well as the `$emp_code`, `$emp_name` and `$emp_designation` properties inherited from the Employee class.
- If a derived class has a property or method with the same name as one in its parent class, the property or method in the derived class takes precedence over, or overrides, the property or method in the parent class. Referencing the property returns the value of the property on the child, while referencing the method calls the method on the child.
- **Note:** Override a method in a subclass when the parent class's implementation is different from that required by the subclass. To access an overridden method on an object's parent class, use the parent::method() notation:

- **Example:**

```
parent::display();
```

- Similarly in the derived class, parent class constructor can be invoked as,

```
parent::__construct();
```

Program 3.6: Write a PHP script to define a class Employee with emp_code, emp_name and emp_designation as data members. Define constructor, destructor and display() as member functions. Derive a class EmployeeSalary from Employee with constructor and member function CalculateSalary() to calculate the net salary of an employee.

```
<?php
class Employee
{
   private $emp_code, $emp_name, $emp_designation;
   function __construct($c,$n,$d)     //constructor definition
   {
      $this->emp_code=$c;
      $this->emp_name=$n;
      $this->emp_designation=$d;
      echo "<br> Object is created";
   }
   function display()
   {
      echo "<br>Emp Code: ".$this->emp_code;
      echo "<br>Emp Name: ".$this->emp_name;
      echo "<br>Emp Designation: ".$this->emp_designation;
   }
   function __destruct()   //destructor definition
   {
      echo "<br>Object is destroyed";
   }
}
class EmployeeSalary extends Employee
{
   private $basic_pay, $earnings, $deduction;
   function EmployeeSalary($c,$n,$d,$b,$e,$dd)
   {
      parent::__construct($c,$n,$d);
      $this->basic_pay=$b;
      $this->earnings=$e;
      $this->deduction=$dd;
   }
```

```
    function CalculateSalary()
    {
        echo "<br> Net Salary(Rs.) :".($this->basic_pay+ $this->earnings-
    $this->deduction);
    }
}
$c=1;
$n="Atharv ";
$d="Manager";
$b=10000;
$e=500;
$dd=750;
$emp=new EmployeeSalary($c,$n,$d,$b,$e,$dd);
$emp->display();
$emp->CalculateSalary();
?>
```

Output:

Object is created
Emp Code: 1
Emp Name: Atharv
Emp Designation: Manager
Net Salary(Rs.) :9750
Object is destroyed

3.6.1 Final Method

- If you declare the method using **final** keyword in superclass/parent class, then subclass/child class can't override that method. Method should not be overridden due to security reason.
- Properties and constants cannot be declared final, only classes and methods may be declared as final.

Program 3.7: Write a PHP script to illustrate the concept of final method.

```
<?php
class A
{
   private $message= "Hi";
   final function display()
   {
         echo $this->message;
   }
```

```
}
class B extends A
{
   function display()
   {
      echo "Hello";
   }
}
?>
```

Output:

Fatal error: Cannot override final method A::display()
n D:\xampp\htdocs\OOP\finalmethod.php on line 12

In the derived class B, display() function can't be redefined as it is declared as final in base class A, so you will get error message.

Program 3.8: Write a PHP script to illustrate the concept of final method.

```
<?php
class A
{
   private $message= "Hi<br>";
   final function display()
   {
         echo $this->message;
   }
}
class B extends A
{
   function displayB()
   {
      echo "Hello";
   }
}
$obj=new B;
$obj->display();
$obj->displayB();
?>
```

Output:

Hi
Hello

- Here $obj, object of class B can invoke display() function of it's base class A without overriding it. Also $obj invokes its own member function display().

3.6.2 Abstract Class and Abstract Method

- Abstraction is a way of hiding information. In abstraction, there should be at least one method that must be declared but not defined.
- Any class that contains at least one abstract method must also be declared as abstract.
- **Syntax :**

```
abstract class classname
{
    //abstract method declaration
    // non-abstract method definition
}
```

Characteristics of abstract class:

- There must be an abstract keyword that must be written before this class for it to be an abstract class.
- This class cannot be instantiated. Only the class that implements all methods of an abstract class can be instantiated. There can be more than one method that can be left undefined.
- If the derived class which inherits the abstract class doesn't define all abstract method then the derived class can't be instantiated.

Program 3.9: Write a PHP script to illustrate the concept of abstract class and abstract method.

```
<?php
abstract class A
{
    abstract function one();
    public function two()
    {
        echo "Non-abstract method";
    }
}
class B extends A
{
    public function one()
    {
        echo "Abstract Function one defined by subclass<br/>";
    }
}
$obj = new B();
$obj->one();
$obj->two();
?>
```

Output:

Abstract Function one defined by subclass.
Non-abstract method.

3.7 INTERFACES

- An Interface allows the users to create programs, specifying the public methods that a class must implement. The interface contains no data variables.
- It does not tell how these methods should be implemented. The method implementation depends upon the class which implements them.

Characteristics of an Interface:

- Interface is similar to a class except that it cannot contain code.
- An interface can define method names and arguments, but not the contents of the methods.
- All methods declared in an interface must be public.
- Any classes implementing an interface must implement all methods defined by the interface.
- A class can implement multiple interfaces.
- An interface is declared using the "interface" keyword.
- Interfaces can't maintain non-abstract methods.

Advantage of an Interface:

- It separates the implementation and defines the structure.
- It is used to define a generic template.

Program 3.10: Write a PHP script to illustrate the concept of interface.

```
<?php
interface myInterface
{
   public function method_one();
   public function method_two();
}
class myClass implements myInterface
{
   public  function method_one()
   {
      echo "<BR> Method one is implemented";
   }
   public  function method_two()
   {
      echo "<BR> Method two is implemented";
   }
}

$obj=new myClass ();
$obj->method_one();
$obj->method_two();
?>
```

Output:

Method one is implemented.

Method two is implemented.

3.8 ENCAPSULATION

- A class is kind of a container or capsule, which contains properties and a set of methods to manipulate these properties. This is called as **Encapsulation.** It binds together code and the data it manipulates.
- The variables or data of a class are hidden from any other class and can be accessed only through any member function of that class. The data in a class is hidden from other classes is called as **Data Hiding**.
- This allows a class to change its internal implementation without affecting overall functionality. The details of implementation of class methods are hidden from the users of the object.
- This is a fundamental concept of object oriented programming. Encapsulation refers to: protection of a class's. Internal data from the code outside that class and the hiding of the details of implementation. Since other classes don't have direct access to the properties, you can change the way the methods get and set the properties.
- Generally, all data members of a class should be declared private. Any access needed to this data by code outside the class should be done through a public method. The protected members can be accessed by methods which are declared as public as well as derived class's public methods.

Advantages of Encapsulation:

1. You can change implementation details at any time without affecting code that uses class.
2. Outside class can't modify property values of an object built from your class without your knowledge.

PROGRAMS

Program 3.11: Write a PHP script to define a class with two numbers as data members and define member's functions to perform arithmetic operations.

```
<?php
class cal
{
    private $no1, $no2;
    function cal($x,$y)
    {
        $this->no1=$x;
        $this->no2=$y;
    }
    function add()
    {
```

```
        return ($this->no1+$this->no2);
    }
    function sub()
    {
        return ($this->no1-$this->no2);
    }
    function mul()
    {
        return ($this->no1*$this->no2);
    }
    function div()
    {
        if($this->no2!=0)
        return ($this->no1)/($this->no2);
    }
}
$x=4;
$y=5;
$obj=new cal(4,5);
echo "Number1= $x <BR> Number2= $y";
echo "<BR> Addition is : ".$obj->add();
echo "<BR> Subtraction is : ".$obj->sub();
echo "<BR> Multiplication is : ".$obj->mul();
echo "<BR> Division is : ".$obj->div();
?>
```

Output:

Number1= 4
Number2= 5
Addition is : 9
Subtraction is : –1
Multiplication is : 20
Division is : 0.8

Program 3.12: Write a PHP script to define an interface which has methods area( ), volume( ). Define constant PI. Create a class Cylinder which implements interface methods and calculate area and volume. Also declare array of objects and call member functions area() and volume().

```
<?php
define('PI', '3.14');

interface Myinterface
```

```
{
    public function area();
    public function volume();
}
class cylinder implements Myinterface
{
    private $r, $h;
    function cylinder($r,$h)
    {
        $this->r=$r;
        $this->h=$h;
    }
    public function area()
    {
        $a=2*PI*$this->r*($this->r+$this->h);
        echo "<br>Area of cylinder= :$a";
    }
    public function volume()
    {
        $v=PI*$this->r*$this->r*$this->h;
        echo "<br>Volume of cylinder= :$v";
    }
}
$r=3;
$h=4;
$obj=new cylinder($r,$h);        //constructor gets invoked for $obj object
echo "<br> Cylinder 1:";
$obj->area();
$obj->volume();
$c[]=array();                    // array of objects
$c[0]=new cylinder(2,3);         //constructor gets invoked for $c[0] object
echo "<br><br> Cylinder 2:";
$c[0]->area();
$c[0]->volume();
$c[1]=new cylinder(6,7);         //constructor gets invoked for $c[1] object
echo "<br><br> Cylinder 3:";
$c[1]->area();
$c[1]->volume();
?>
```

Output:

```
Cylinder 1:
Area of cylinder= :131.88
Volume of cylinder= :113.04

Cylinder 2:
Area of cylinder= :62.8
Volume of cylinder= :37.68

Cylinder 3:
Area of cylinder= :489.84
Volume of cylinder= :791.28
```

- In above PHP script, `$obj` is initialized by calling constructor as follows:
  `$obj=new cylinder($r,$h);`
- The following statement initializes object from an array:
  `$c[0]=new cylinder(2,3);`
- The objects `$c[0]` and `$c[1]` invokes `area()` and `volume()` functions using `->` operator. The `area()` and `volume()` functions uses respective object's data members for computation of area and volume.

Summary

- PHP supports object oriented concepts. Class can be defined using a keyword class. Using new keyword, object can be declared.
- Using object, we can access its properties and methods using -> operator.
- The constructor is used to initialize the object and destructor is used release the memory when it goes out of scope.
- We also learned about different functions related to observing characteristics of an object and a class.
- Use `serialize()` and `unserialize()` functions to convert object to byte stream and byte stream to object respectively.
- Using extends keyword, the properties and methods of one class can be inherited into another class. Also interface can be written using keyword interface, which can be implemented by a class.
- In abstraction, there should be at least one method that must be declared but not defined. The class that inherits this abstract class needs to define that method.
- PHP also supports to define a class with static property and static method.
- The ability to hide the details of implementation of class methods from the users of the object is called as Encapsulation.
- All data members of a class should be declared private. Any access needed to those variables by code outside the class should be done through a public method.

Check Your Understanding

Q.I Multiple choice questions.

1. Which notation is used to access variables of an object?
   (a) :: (b) =
   (c) -> (d) .
2. Which notation should be used to refer to a method in the context of a class rather than an object you use?
   (a) -> (b) _
   (c) $ (d) ::
3. Which method is invoked just before an object is garbage collected?
   (a) _collect() (b) _garbage()
   (c) _destruct( ) (d) _destructor()
4. Which one of the following property scopes is not supported by PHP?
   (a) friendly( ) (b) final
   (c) public (d) static
5. Which function is used to check if class exists or not?
   (a) exist() (b) exist_class()
   (c) class_exists( ) (d) _exist()
6. Which keyword is used to refer to properties or methods within the class itself?
   (a) private (b) public
   (c) protected (d) $this
7. Objects are also known as _____.
   (a) reference (b) class
   (c) instance (d) template
8. A member function typically accesses members of _____ object only.
   (a) current (b) next
   (c) previous (d) all of the above
9. The practice of separating the user from the true inner workings of an application through well-known interfaces is known as_____.
   (a) Polymorphism (b) Inheritance
   (c) Encapsulation (d) Dynamic Message passing
10. PHP recognizes constructor by the name _____.
   (a) classname() (b) __constrct
   (c) function _construct() (d) function __construct()

11. Which one of the following can be used to instantiate an object in PHP assuming class name to be foo?
    (a) $obj = new $foo;
    (b) obj = new foo;
    (c) $obj = new foo ();
    (d) obj = new foo();
12. Which one of the following is the right way to invoke a method?
    (a) $object->methodName();
    (b) object->methodName();
    (c) object->methodName;
    (d) $object::methodName;

ANSWERS

1. (c)	2. (d)	3. (c)	4. (a)	5. (c)	6. (d)	7. (c)	8. (a)
9. (c)	10. (d)	11. (c)	12. (a)				

Q.II State True/False.

1. PHP supports static methods.
2. exist_class() function is used to check if class exists or not.
3. The use of $this in function definition is optional.
4. PHP does not support anonymous functions.
5. An abstract class can't be instantiated.
6. All methods declared in an interface must be public.
7. Any class that contains at least one abstract method must also be declared as abstract.
8. Destructor must takes at least one parameter.
9. To access parent class method self keyword is used.
10. The conversion of any PHP value to a byte stream representation is called as Serialization.

ANSWERS

1. True	2. False	3. False	4. False	5. True
6. True	7. True	8. False	9. False	10. True

Practice Questions

Q. I Answer the following questions in short.

1. State the purpose of extends keyword?
2. How to create object in PHP ?
3. Which function is used to check if class is exists or not?
4. State the purpose of self keyword?

5. How static method is invoked?
6. What is the use of abstract class?
7. State the purpose of parent keyword?
8. Write characteristics of abstract class.
9. Write characteristics of an interface.
10. What is serialization?

Q. II Answer the following questions.

1. How to call a constructor of a parent class from a child class? Explain with suitable example.
2. What is Inheritance? Explain with suitable example.
3. Explain the concept of static methods and static property with suitable example.
4. What is an Interface? Illustrate the concept with suitable example.
5. What is abstract class and abstract method? Illustrate with suitable example.
6. What is Method Overriding? Explain with suitable example.
7. Explain different functions with syntax to examine characteristics of an object and a class.
8. Write a PHP Script to create a class Worker that has data members as `Worker_Name`, `No_of_Days_worked`, `Pay_Rate`. Define parameterized constructor. Also write necessary member function to calculate and display the salary of worker.
9. Write a PHP program to create a class temperature which contains data members as Celsius and Fahrenheit. Define parameterized constructor to initialize all values of temperature object. Convert Celsius to Fahrenheit and Convert Fahrenheit to Celsius using member functions. Display conversion on next page.
10. Define a class Employee having private members – id, name, department, salary. Define parameterized constructors. Create a subclass called "Manager" with private member bonus. Create 3 objects of the Manager class and display the details of the manager having the maximum total salary (salary + bonus).

Q.III Define the terms

1. Class
2. Object
3. Encapsulation
4. Constructor
5. Destructor
6. Inheritance
7. Method Overriding
8. Final method
9. Abstract class

***

4... Web Techniques

Objectives...

- To understand server information.
- To study about processing forms.
- To learn how to use sticky forms.
- To study how to set response headers.
- To know about how to maintain state and SSL.

4.1 INTRODUCTION

- Form processing is essential component in almost every web applications, through forms as interface user communicates with server.
- In PHP program, form is used to display the form. It involves HTML tags such as text boxes, checkbox, radio buttons, buttons etc. After creating a form you can insert data into it and that information is sent to the server which will then pass on to database and it gets stored into it. Another way is to pull the data from the database and display on to form, to retain information on to the form we use the concept of sticky forms.
- Along with forms we use GET, POST variables to send the data to the server. Similarly, $_SERVER[] global array is also used for server information; it is used for setting response headers of page.

4.2 VARIABLES

- PHP creates a separate global variable for every form parameter, for every piece of request information, and every server configuration value.
- PHP creates six global arrays that contain the EGPCS (Environment, GET, POST, Cookie and Server) information.

(1) **$_COOKIE** : It contains any cookie value passed as part of request, where the keys of the array are the names of the cookies. It is an associative array of variables passed to the current script via HTTP Cookies.

(2) **$_GET**: It contains any parameters that are part of GET request, where the keys of the array are the names of the form parameters.

(3) **$_POST:** It contains any parameters that are part of a POST request, where the keys of the array are the names of the form parameters.

(4) **$_SERVER:** It contains useful information about the web server, $_GET and $_POST and $_SERVER super global array variables we have discussed in detail in next point.

(5) **$_FILES:** It contains information about any uploaded files. It is a two dimensional associative global array of items which are being uploaded by via HTTP POST method and holds the attributes of the files such as,

Table 4.1: Attributes of $_FILES

Attribute	Description
name	name of the file which is uploading.
size	size of the file.
type	type of the file.
tmp_name	a temporary address where the file is located before processing upload request.
Error	type of error while file is uploading.

(6) **$_ENV** : $_ENV is superglobal associative array in PHP. It stores environment variables available to current script. $HTTP_ENV_VARS also contains the same information, but is not a superglobal, and now been deprecated. Environment variables are imported into global namespace. Most of these variables are provided by the shell under which PHP parser is running. Hence, list of environment variables may be different on different platforms. This array also includes CGI variables in case whether PHP is running as a server module or CGI processor.

Program 4.1: PHP program to demonstrate use of cookie.

```
<?php
setcookie("user_name", "TestUser", time()+ 60,'/'); // expires after 60
seconds
echo 'the cookie has been set for 60 seconds'.'<br>';
print_r($_COOKIE);
?>
```

Output:

```
the cookie has been set for 60 seconds
Array ( [user_name] => TestUser ) ?>
```

Program 4.2: PHP program to demonstrate use of $_FILES.

```
<?php
echo "<pre>";
print_r($_FILES);
echo "</pre>";
?>
<form action="" method="post" enctype="multipart/form-data">
<input type="file" name="file">
<input type="submit" value="Upload Image">
</form>
```

Output:

```
Array
(
    [file] => Array
        (
            [name] => tableimg.odg
            [type] => application/vnd.oasis.opendocument.graphics
            [tmp_name] => /tmp/phpeG9tLC
            [error] => 0
            [size] => 293298
        )

)
```

Browse... No file selected. Upload Image

Program 4.3: PHP program to demonstrate use of $_ENV variable.

```
<?php
echo 'My username is ' .$_ENV["USER"] . '!';
?>
```

Output:

My username is suresh.

4.3 SERVER INFORMATION

- In addition to PHP_SELF and REQUEST_METHOD, the $_SERVER is an auto global array it contains number of useful elements that provide information on to the web server and current request.
- $_SERVER array contains useful information from the web server. $_SERVER is a super global variable it contains information about headers, path, host and script locations.
- With these global indices, the $_SERVER array contains values for the meta variable listed with the specification of CGI (Common Gateway Interface).

- Following is the list of entries in $_SERVER that comes from CGI.

Table 4.2: $_SERVER Elements

Sr. No.	Element/Code	Description
1.	$_SERVER['PHP_SELF']	Returns the filename of the currently executing script.
2.	$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using.
3.	$_SERVER['SERVER_ADDR']	Returns the IP address of the host server.
4.	$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.nirali.com).
5.	$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24).
6.	$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1).
7.	$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST).
8.	$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496).
9.	$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string.
10.	$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request.
11.	$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8, ISO-8859-1).
12.	$_SERVER['HTTP_HOST']	Returns the Host header from the current request.
13.	$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it).
14.	$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol?
15.	$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page.
16.	$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page.

Sr. No.	Element/Code	Description
17.	$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server.
18.	$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script.
19.	$_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).
20.	$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80).
21.	$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages.
22.	$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script.
23.	$_SERVER['SCRIPT_NAME']	Returns the path of the current script.
24.	$_SERVER['SCRIPT_URI']	Returns the URI of the current page.

- The Apache server also creates entries in the $_SERVER array for each HTTP header in the request. For each key, the header name is converted to uppercase, hyphens (-) are turned into underscores

  (_), and the string "HTTP_" is prepended. For example, the entry for the User-Agent header has the key "HTTP_USER_AGENT".
- The two most common and useful headers are:

1. HTTP_USER_AGENT: The string the browser used to identify itself (For example, "Mozilla/5.0 (Windows 2000;U) Opera 6.0 [en]").
2. HTTP REFFERER: The page the browser said it came from to get to the current page (For example, "http://www.example.com/last_page.html").

Program 4.4: PHP program to demonstrate $_SERVER information.

```
<!DOCTYPE html>
<html>
<body>
<?php
```

```
echo "php self=".$_SERVER['PHP_SELF'];
echo "<br>";
echo "server name=".$_SERVER['SERVER_NAME'];
echo "<br>";
echo "server host=".$_SERVER['HTTP_HOST'];
echo "<br>";
echo "http user agent=".$_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo "script name=".$_SERVER['SCRIPT_NAME'];
echo "<br>";
echo "server software".$_SERVER['SERVER_SOFTWARE'];
echo "<br>";
echo "gateway interface=".$_SERVER['GATEWAY_INTERFACE'];
echo "<br>";
echo "server protocol".$_SERVER['SERVER_PROTOCOL'];
echo "<br>";
echo "server port=".$_SERVER['SERVER_PORT'];
echo "<br>";
echo "remote addr=".$_SERVER['REMOTE_ADDR'];
?>
</body>
</html>
```

Output:

```
php self=/phpexamples/phpserver.php
server name=localhost
server host=localhost
http user agent=Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
script name=/phpexamples/phpserver.php
server softwareApache/2.4.41 (Ubuntu)
gateway interface=CGI/1.1
server protocolHTTP/1.1
server port=80
remote addr=127.0.0.1
```

Program 4.5: PHP program to display all the $_SERVER parameters.

```
<?php
foreach ($_SERVER as $parm => $value)
echo "<br>"."$parm = '$value'\n";
?>
```

Output:

```
HTTP_HOST = 'localhost'
HTTP_USER_AGENT = 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0'
HTTP_ACCEPT = 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8'
HTTP_ACCEPT_LANGUAGE = 'en-US,en;q=0.5'
HTTP_ACCEPT_ENCODING = 'gzip, deflate'
HTTP_CONNECTION = 'keep-alive'
HTTP_UPGRADE_INSECURE_REQUESTS = '1'
HTTP_CACHE_CONTROL = 'max-age=0'
PATH = '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin'
SERVER_SIGNATURE = '
Apache/2.4.41 (Ubuntu) Server at localhost Port 80
'
SERVER_SOFTWARE = 'Apache/2.4.41 (Ubuntu)'
SERVER_NAME = 'localhost'
SERVER_ADDR = '127.0.0.1'
SERVER_PORT = '80'
REMOTE_ADDR = '127.0.0.1'
DOCUMENT_ROOT = '/var/www/html'
REQUEST_SCHEME = 'http'
CONTEXT_PREFIX = ''
CONTEXT_DOCUMENT_ROOT = '/var/www/html'
SERVER_ADMIN = 'webmaster@localhost'
SCRIPT_FILENAME = '/var/www/html/phpexamples/serverentries.php'
REMOTE_PORT = '47756'
GATEWAY_INTERFACE = 'CGI/1.1'
SERVER_PROTOCOL = 'HTTP/1.1'
REQUEST_METHOD = 'GET'
QUERY_STRING = ''
REQUEST_URI = '/phpexamples/serverentries.php'
SCRIPT_NAME = '/phpexamples/serverentries.php'
PHP_SELF = '/phpexamples/serverentries.php'
REQUEST_TIME_FLOAT = '1606737480.339'
REQUEST_TIME = '1606737480'
```

4.4 PROCESSING FORMS

- Forms are essential parts in web development. Forms are used to communicate between users and the server.
- Form is a used to get information of the user to the server and let the server do something in response to the user's input.
- It's very easy to process forms with PHP, you can process information collected by HTML form and you can use PHP code to make decisions based on this information to create dynamic web pages.
- Before you process the information, you need to create an HTML form that will send information to PHP script.
- There are two ways for sending data from web page using GET and POST.
- GET and POST are defined in form element's method attribute. Along with this you need to specify action attribute with the name of the file.

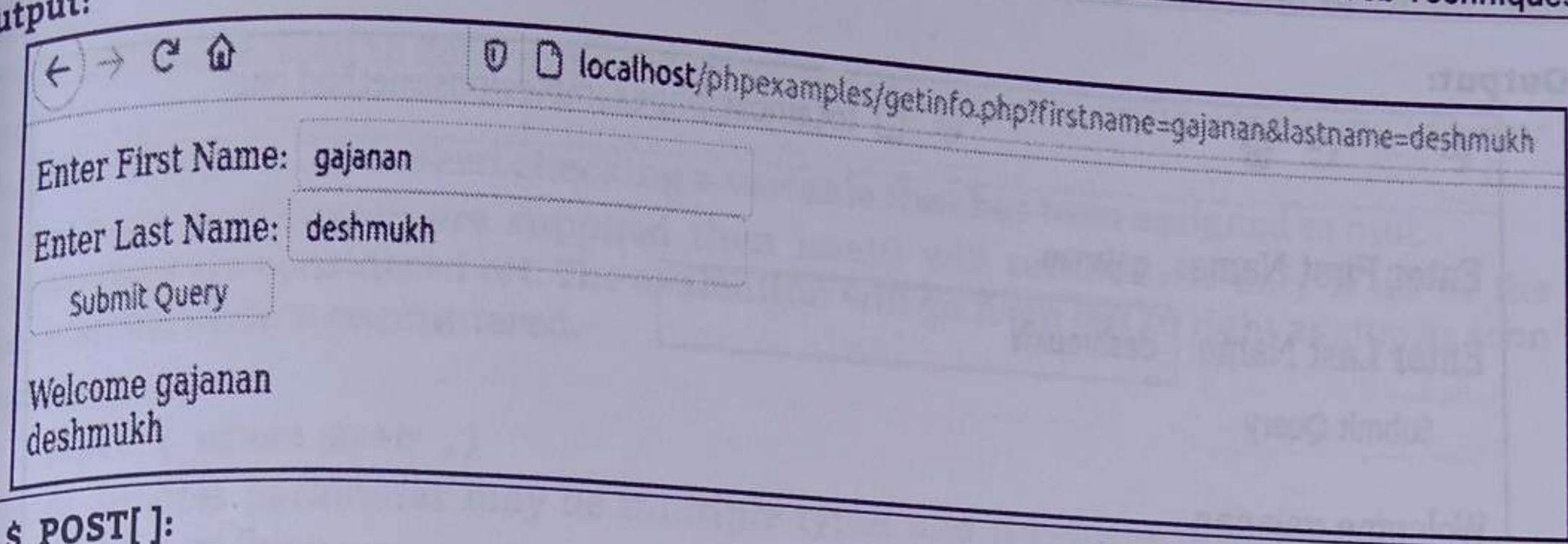
1. $_GET[ ]:

- It is used to retrieve information from the form control through parameter sent in the URL. The page and the encoded information are separated by the ? character.
- It is an associative array of variables passed to the current script via the URL parameters. You can use when there is small amount of data, it is mostly used in pagination, page number is shown in the URL and you can easily get the page number from URL using $_GET.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- The GET method is restricted to send up to 1024 characters only. GET cannot 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.

Program 4.6: PHP program to demonstrate $_GET[] in PHP.

```
<?php
if( $_GET["firstname"] || $_GET["lastname"] )
{
    echo "Welcome ". $_GET['firstname']. "<br />";
    echo $_GET['lastname'];
    exit();
}
?>
<html>
<body>
<form action = "<?php $_PHP_SELF ?>" method = "GET">
Enter First Name: <input type = "text" name = "firstname" /><br>
Enter Last Name: <input type = "text" name = "lastname" /><br>
<input type = "submit" />
</form>
</body>
</html>
```

Output:



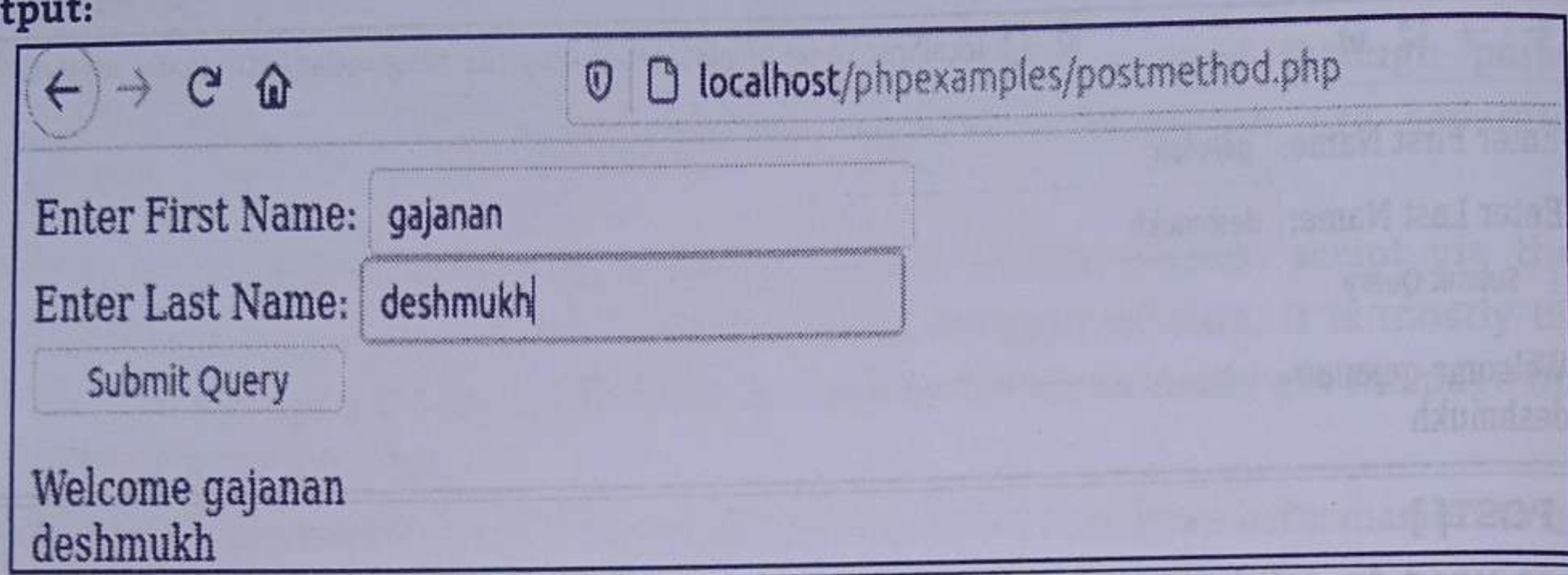
2. $_POST[ ]:

- It is used to retrieve the information from the form control through HTTP POST method.
- It is an associative array of variables passed to the current script via the HTTP POST method.
- You can use POST method when you are sending large data to server or if you have sensitive information like passwords, credit card details etc.
- Unlike the GET method, it does not have a limit on the amount of information to be sent. The information sent from an HTML form using the POST method is not visible to anyone.

Program 4.7: PHP program to demonstrate $_POST[ ] method.

```
<?php
if( $_POST["firstname"] || $_POST["lastname"] )
{
   echo "Welcome ". $_POST['firstname']. "<br/>";
   echo $_POST['lastname'];
   exit();
}
?>
<html>
<body>
<form action = "<?php $_PHP_SELF ?>" method = "POST">
   Enter First Name: <input type = "text" name = "firstname" /><br>
   Enter Last Name: <input type = "text" name = "lastname" /><br>
<input type = "submit" />
</form>
</body>
</html>
```

Output:



- **Difference between GET and POST Method:**

Table 4.3: Difference between GET and POST

GET Method	POST Method
1. The GET method sends its variables in the web browsers URL which makes it easy to see and possibly change the information that was sent.	1. The POST method sends variables securely and it has no limitations on the amount of data to be sent. And variables are not displayed in URL. Also it is not possible to bookmark the page.
2. So this method should not be used when sending passwords or other sensitive information. It also should not be used for any actions that cause a change in the server, such as placing an order or updating a database. However, because the variables are displayed in the URL; it is possible to bookmark the page.	2. The biggest difference between GET and POST is that GET requests are idempotent, GET request for a particular URL, including form parameters, is the same as two or more requests for that URL. Thus, web browsers can cache the response pages for GET requests, because the response page doesn't change regardless of how many times the page is loaded. Because of idempotence.
3. The GET method has limit on the amount of data to be sent. If you send long variables using GET, you are likely to lose large amount of data. Another point is all the variables which you pass are visible in URL separating with ? So GET method is not secure.	3. POST requests are not idempotent. This means that they cannot be cached, and the server is recontacted every time the page is displayed.
4. `Form name="form1" method="GET"`	4. `Form name="form1" method="POST"`

- Apart from `$_GET[]` and `$_POST[]` super global array; following methods are also used to process forms.

3. **isset():**

- This function is used to determine whether variable or form control is having value or not.
- isset() will return false when checking a variable that has been assigned to null.
- If multiple parameters are supplied then isset() will return true only if all of the parameters are considered set. The evaluation will go from left to right as stop as soon as unset variable is encountered.
- **Syntax:**

```
isset ( mixed $var ,)
```

- mixed denotes parameter may be multiple types and it returns boolean value which may be true or false

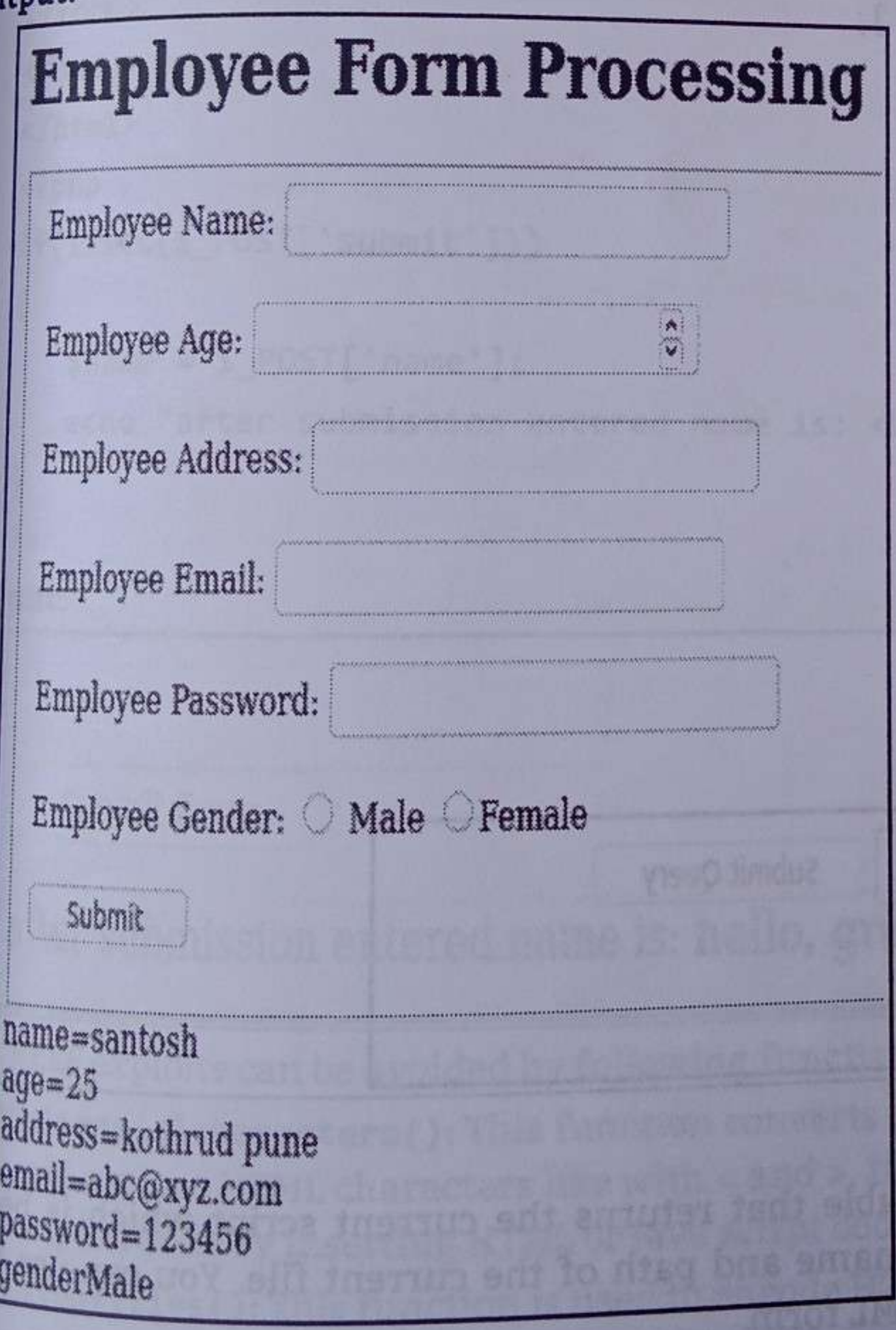
Program 4.8: PHP program to demonstrate form processing using isset() in PHP :

```
<?php
if (isset($_POST['submit']))
{
  if ((!isset($_POST['ename'])) || (!isset($_POST['age'])) ||
  (!isset($_POST['address'])) || (!isset($_POST['emailaddress'])) ||
  (!isset($_POST['password'])) || (!isset($_POST['gender'])))
  {
    $error = "*" . "Please fill all the required fields";
  }
  else
  {
    $name = $_POST['ename'];
    $age = $_POST['age'];
    $address = $_POST['address'];
    $emailaddress = $_POST['emailaddress'];
    $password = $_POST['password'];
    $gender = $_POST['gender'];
  }
}
?>
<html>
<head>
<title>Employee Form Processing</title>
</head>
<body>
<h1>Employee Form Processing </h1>
<fieldset>
```

```
<form id="form1" method="post" action="">
<?php
if (isset($_POST['submit']))
{
    if (isset($error))
    {
        echo "<p style='color:red;'>"
        . $error . "</p>";
    }
}
?>
Employee Name:
<input type="text" name="ename"/>
<br>
<br>
Employee Age:
<input type="number" name="age"/>
<br>
<br>
Employee Address:
<input type="text" name="address"/>
<br>
<br>
Employee Email:
<input type="email" name="emailaddress"/>
<br>
<br>
Employee Password:
<input type="password" name="password"/>
<br>
<br>
Employee Gender:
<input type="radio"value="Male"
name="gender"> Male
<input type="radio" value="Female"
name="gender">Female
<br>
<br>
<input type="submit" value="Submit" name="submit" />
</form>
</fieldset>
<?php
```

```
if(isset($_POST['submit']))
{
    if(!isset($error))
    {
        echo "name=".$name."<br>";
        echo "age=".$age."<br>";
        echo "address=".$address."<br>";
        echo "email=".$emailaddress."<br>";
        echo "password=".$password."<br>";
        echo "gender".$gender."<br>";
    }
}
?>
</body>
</html>
```

Output:



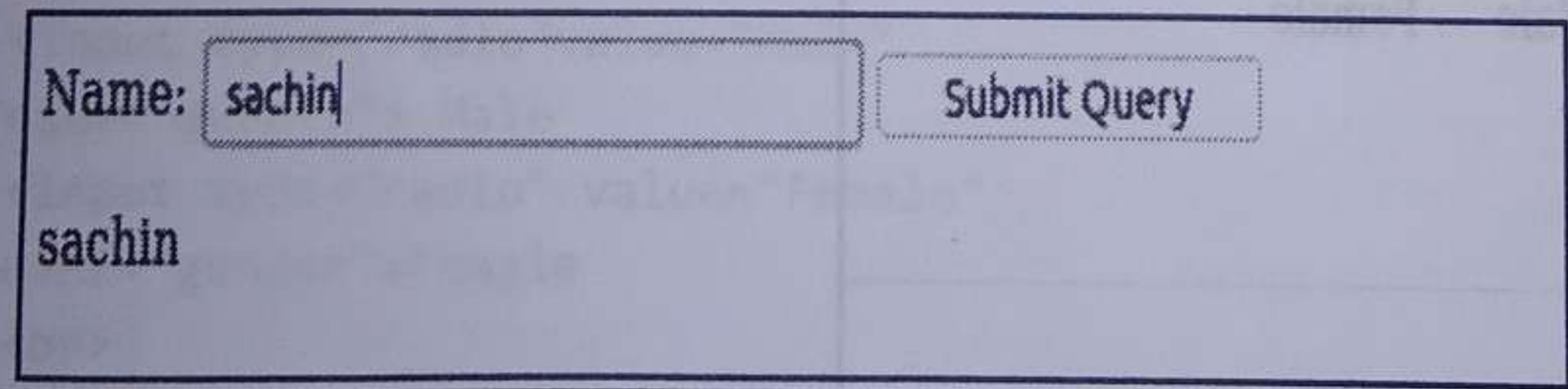
4. $_REQUEST[]:

- PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.
- It is also used to access information while using database.

Program 4.9: PHP program to demonstrate $_REQUEST[].

```
<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")
{
    $name = $_REQUEST['fname'];
    if (empty($name))
    {
        echo "Name is empty";
    }
    else
    {
        echo $name;
    }
}
?>
</body>
</html>
```

Output:



4.4.1 Self Processing Pages

- PHP_SELF is a super global variable that returns the current script which is being executed. This variable returns name and path of the current file. You can use this variable in action field of the HTML form.

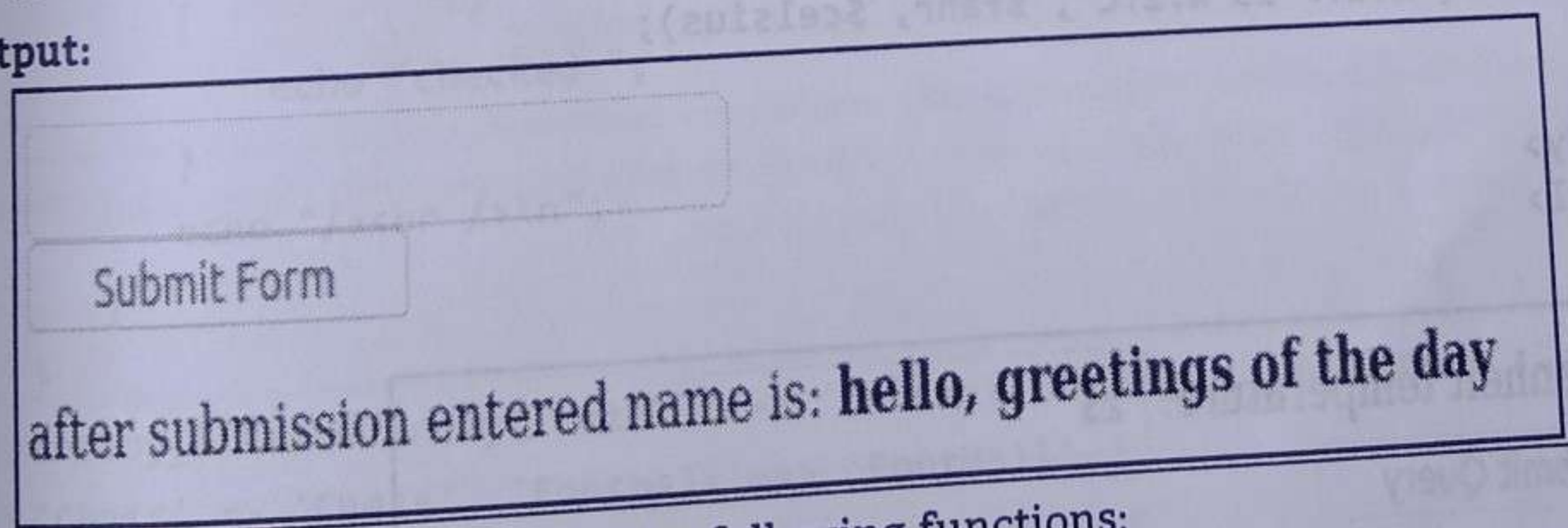
- The most common use of PHP_SELF variable is in the action field of the form tag `<form action="">`. Action field instructs where to submit data when user clicks on submit button. Assume you have file name called `php_example.php` and you want to load the same page after the form submission.
- Then code will be, `<form method="post" action="php_example.php">`
- We can also use PHP_SELF variable instead of `php_example.php`, so the code becomes like this,

```
<form name="form1" method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

Program 4.10: PHP program to demonstrate PHP_SELF.

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
<input type="text" name="name"><br>
<input type="submit" name="submit" value="Submit Form"><br>
</form>
</body>
</html>
<?php
if(isset($_POST['submit']))
{
    $name = $_POST['name'];
    echo "after submission entered name is: <b> $name </b>";
}
?>
```

Output:



PHP_SELF exploits can be avoided by following functions:

1. `htmlspecialcharacters()`: This function converts special characters to HTML entities. It will replace HTML characters like with < and >. It prevents scripting attacks who try to exploit code by inserting HTML or Java script code.
2. `htmlentities()`: This function is used to encode HTML entities. Now if the user try's to exploit the PHP_SELF variable the attempt will fail.

4.4.2 Sticky Forms

- When you are working with forms on the web page you may come to know that after submitting the information when you want to go back and correct it you may see an empty form.
- If you are sent back to the form that retains the information which already entered you are supposed to be using sticky forms.
- A sticky form is simply HTML form that remembers how you filled it out. It is nice feature; it reduces efforts of user to reenter data into form.
- Most of the web sites uses sticky form concept in which results of query are accompanied by a search form whose default values are those of previous query.

Program 4.11: Program of converting temperature from Fahrenheit to celcius.

```
<html>
<head><title>Temperature Conversion</title></head>
<body>
<?php
$fahr = $_GET['fahrenheit'];
?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Fahrenheit temperature:
<input type="text" name="fahrenheit" value="<?php echo $fahr ?>" />
<br />
<input type="submit" name="Convert to Celsius!" />
</form>
<?php
if (! is_null($fahr))
{
   $celsius = ($fahr - 32) * 5/9;
   printf("%.2fF is %.2fC", $fahr, $celsius);
}
?>
</body>
</html>
```

Output:

Fahrenheit temperature: 25

Submit Query

25.00F is -3.89C

- In the above example, sticky form is used with `value="<?php echo $fahr ?>"` here after submitting the form data the input value will remain visible onto form.

4.4.3 Sticky Multivalued Parameters

- Now, you may think can we use sticky multiple form elements? The answer is yes, but you need to check whether each possible value in the form was one of the submitted value.

Program 4.12: PHP program for multivalued checkbox.

```
<html>
<head><title>Hobbies</title></head>
<body>
<?php
// fetch form values, if any
$attrs = $_GET['attributes'];
if (! is_array($attrs))
{
   $attrs = array( );
}
// create HTML for identically-named checkboxes
functionmake_checkboxes ($name, $query, $options)
{
   foreach ($options as $value => $label)
   {
      printf('%s <input type="checkbox" name="%s[]" value="%s" ',
      $label, $name, $value);
      if (in_array($value, $query))
      {
         echo "checked ";
      }
      echo "/><br />\n";
   }
}
$hobby_likes = array('cricket'=> 'cricket', 'Tennis'=> 'Tennis',
'Chess' => 'Chess', 'Football' => 'Football',
'Swimming' => 'Swimming', 'Snooker' => 'Snooker'
);
?>
<form action="<?php $_SERVER['PHP_SELF'] ?>" method="GET">
Select your hobbies:<br />
```

```
<?phpmake_checkboxes('attributes', $attrs, $hobby_likes); ?>
<br />
<input type="submit" name="s" value="Select your hobbies!" />
</form>
<?php
if (array_key_exists('s', $_GET))
{
    $likes = join (" ", $_GET['attributes']);
    echo "You have selected $likes as your hobbies.";
}
?>
</body>
</html>
```

Output:

Select your hobbies:
cricket ☑
Tennis ☑
Chess ☑
Football ☐
Swimming ☐
Snooker ☑

Select your hobbies!

You have selected cricket Tennis Chess Snooker as your hobbies.

- In above program, `make_checkboxes()` function has passed with three arguments which is used to set sticky multivalued checkboxes.

4.5 SETTING RESPONSE HEADERS

- The HTTP response that a server sends back to the client contains header that identify the type of content in the body of the response. The server that sends response, how many bytes are in the body, when the response was sent etc. The Servers Apache and PHP take care of headers, identifying the document as HTML, calculating the length of HTML page, and so on.
- You can set the expiration time of page, redirect the client browser, or to generate the HTTP error, you can use `header()` function.
- Headers must be written at the top of the file before HTML tags.

- **Content-type:** The content type header identifies the type of the document which will get returned. It is normally text/html but you can specify text/plain also.

```
<?php
header('content-type:text/plain');
?>
```

- **Redirections:** Redirections will send the browser to the new URL.

```
<?php
header('location:"https://www.google.com");
?>
```

- If there is a partial URL like `"../sample.html"` the redirection is handled by internally by web server. Mostly we should specify absolute URL.
- **Expiration:** Server explicitly informs browser when document will get expire. Proxy and browser cache hold document until the time or it gets expire.
- Expires header is used to set the expiration time:

```
header( 'Expires:  Mon, 20 Jan 2020 06:00:00GMT');
```

- To expire the web page before five hours of page generation time, use `time()` method and `gmstrftime()` to generate expiration string:

```
$now = time();
$string = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 60*60*5);
header("Expires: $string");
```

- If you wish that your document never gets expire then use time and `gmstrftime()`:

```
$now = time( );
$string = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 365*86440);
header("Expires: $string");
```

- This is the best way to prevent a browser or proxy cache from storing your document:

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
```

- The `header()` function is the built-in function in PHP used to send raw HTTP header. These are functions which manipulates information sent to the client by web browser before any output has been sent.
- The PHP `header()` function send a HTTP header to client or browser in a raw form, this raw data is sent with request made by server as header information.
- **Syntax:** `void header( $header, $replace=TRUE , $http_response_code)`
- This header function has three parameters `$header` parameter holds string. There are two types of header calls. The first header starts with $HTTP which is used to check HTTP status code to send. Another is location, it is compulsory parameter.

- `$replace` parameter is used to replace previous header or add second header. The default value is true. If `$replace` is false value then it force multiple headers of same type.
- **`$http_response_code`:** It is optional parameter. It forces the HTTP response code to the specified value.

Program 4.13: PHP program for header function.

```
<?php
// PHP program to describes header function
// Redirect the browser
header("Location: https://www.google.com");
// The below code does not get executed
// while redirecting
exit;
?>
```

Output:

This will change location of header, and you will be redirected to URL given.

Program 4.14: PHP program to display header list.

```
<?php
header("Expires: Wed, 11 Mar 2020 10:04:20 GMT");
header("Cache-Control: no-cache");
header("Pragma: no-cache");
?>
<html>
<body>
<p>Code for php header!</p>
<!-- PHP program to display header list -->
<?php
print_r(headers_list());
?>
</body>
</html>
```

Output:

code for php header!

Array[0]=> Expires: Wed, 11 Mar 2020 10:04:20 GMT [1]=> Cache-control: no-cache[2]=>pragma:no-cache

- The above example helps to prevent caching by sending header information which override browser setting to not-cache.
- Note that only one header is allowed to send at a time to prevent header injection attacks.

HTTP Authentication:

- If you want PHP to protect parts of your web site with password instead of storing in external file you can use `$_SERVER['PHP_AUTH_USER']` and `$_SERVER['PHP_AUTH_PASSWORD']` global variables that contains username and password.
- If you want to deny access to page, you have to send www-Authenticate header which identifies the authentication realm as part of response with status code 401.

```
header('www-Authenticate : Basic realm="website name"');
header('HTTP/1.0 404 Unauthorized Access');
echo "enter valid username and password";
exit;
```

Program 4.15: Basic HTTP Authentication example.

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER']))
{
    header("WWW-Authenticate: Basic realm=\"Private Area\"");
    header("HTTP/1.0 401 Unauthorized");
    // only reached if authentication fails
    print "Sorry - you need valid credentials granted access to the private
    area!\n";
    exit;
}
else
{
    // only reached if authentication succeeds
    print "Welcome to the private area, {$_SERVER['PHP_AUTH_USER']} - you
    used {$_SERVER['PHP_AUTH_PW']} as your password.";
}
?>
```

Output:

Welcome to the private area, gd4184 - you used password as your password.

4.6 MAINTAINING STATE

- HTTP is a stateless protocol. It means that when web servers complete the request of client connection goes away. It means that there is no way to identify that sequence of requests all coming from same client.
- To keep track of state information between requests and to maintain state cookies are used.

- A cookie is a bit of information that server can give to client. On every subsequent request the client will give information back to server, thus identifying itself.
- Cookies are useful for retaining information through repeated visits by browser, but many browsers allow users to disable cookies. So the best way to maintain state with PHP is to use built-in session tracking mechanism.
- It allows you to create persistent variables that are accessible from different pages of your application, as well as different visits to the site by the same user. Behind session uses cookies to solve most problems that require state, taking care of all the details for you.

Cookie:

- It is basically a string that contains several fields. A server can send one or more cookies to a browser in the header of a response.
- Simply we can say cookie is text file saved to user's system by a web site. This file contains information that the site can retrieve on the user's next visit, so that it an allow site to recognize user and it can provide enhanced set of features customized to that user.
- Cookies can have some important security features as,
  - A cookie can only be read by the web site or domain that created it.
  - A single domain can not set more than 20 cookies.
  - A single cookie can not exceed 4KB in size.
- The maximum number of cookies that may be set on user's machine is 300.
- Cookies are managed through browser setting's you can turn on or turn off. Cookies can be misused but mostly they are used for enhancing user experience for a web site.
- Cookies can be created with the function setcookie();
- **Syntax:**

```
setcookie(name,value, [expiry_time],[cooke_path],[domain],[secure]);
```

Cookie Attributes:

Table 4.4: Cookie Attributes

Attribute	Description
name=value	A cookie name and value
expires	The validity of the cookie
domain	The domain associated with the cookie
Path	The domain path associated with the cookie
secure	If present, a secure HTTP connection is required to read the cookie

Creating cookies:

- We will create a cookie to store age of user, its value and set expiry time of one week.

```
<?php
setcookie("name","test","time()+60*60*24*7");
```

- To access the store cookie we can use `$_COOKIE` global variable using `isset()` method to check cookie is set or not.

Program 4.16: Program for creating a cookie.

```
<?php
$exp=time()+86400;          //cookie is valid for one day

setcookie("age","25",$exp);
print_r($_COOKIE);
echo "<br>";
if(isset($_COOKIE['age']))
{
echo "cookie is set";
   }
else
{
echo "cookie is not set";
}
?>
```

Output:

```
Array ( [age] => 25 )
cookie is set
```

Updating Cookie: If you want to update cookie you need to set it again, sometimes you need to modify its username or value you can do it with setcookie() method again.

Program 4.17: Program to demonstrate setcookie() method.

```
<?php
$exp=time()+60*60*24*7;
setcookie("name","suresh",$exp);
if(isset($_COOKIE['age']))
{
echo "cookie is set" . $_COOKIE['name'];
   }
else
{
echo "cookie is not set";
}
?>
```

Output:

Cookies	Name	Value	Domain	Path	Expires / Max-Age	Size
http://localhost	age	25	localhost	/phpexamples	Tue, 07 Jul 2020 04:38:17 GMT	5
	name	suresh	localhost	/phpexamples	Mon, 13 Jul 2020 04:48:16 GMT	10
Indexed DB						

- As you can see in picture we have created two cookies on local server localhost; you can see expiry time also.

Removing Cookies: To Delete or remove cookies again we have to use setcooke() with its name to set expiry date which we need to set in the past with negative value.

Program 4.18: Program for $exp_unset.

```
<?php
$exp_unset=time()-68400;
setcookie("name","suresh",$exp_unset);
if(isset($_COOKIE['name']))
{
   echo "cookie is set" . $_COOKIE['name'];
}
else
{
   echo "cookie is not set";
}
?>
```

Output:

cookie is not set

Sessions:

- PHP has built-in support for sessions. Session allows you to easily create multipage forms (e.g. shopping cart) allowing you to save user authentication information from page to page, and store persistent user preference on site.
- Each first time user is given with unique session ID. By default, the Session ID is stored in cookie called PHPSESSID.
- If the browser does not support cookie or it has turn off, the session ID is propagated in URL's within the website.
- Sessions are started automatically when a script begins running.
- Sessions work very similar to cookies except that cookies are stored at client side and sessions are stored at server side. Session ID can be stored in cookie.
- An unique ID is assigned to user. The most common form of session usage is for commerce sites and the ability to have a shopping cart, user login and customized interfaces, and navigation history.

Creating Sessions:

- First you need to call function `session_start()` which will create new session and to generate session id for the client.
- This needs to be at the top of the PHP page.

- Once session is created, we can attach any number of session variables to it. They act like normal variables and we can store any values into it and this will be exist till the duration of session.
- When the user will close the browser it will get expire. Session variables are created like key value pairs.

Session variables:

- Variables are assigned to session using $_SESSION array. This is a global array that is accessible to all pages of your web site. It is an associative array.
- We will create a session and write session variable name and try to echo it.

```
$_SESSION['name']="suresh";
```

- We can display session id on screen using method session_id();

Program 4.19: PHP program to create session.

```
<?php
session_start();
$_SESSION['name']="suresh";
echo "hello".$_SESSION['name'];
echo session_id();
?>
```

Output:

```
hellosuresh
session id is=b5ivds1q9j2tvguf63t1davk91
```

- Now we will create another file and we will access session variable name into it.

Program 4.20: PHP program to access session variable name.

```
<?php
session_start();
echo "hello welcome back".$_SESSION['name'];
?>
```

Output:

```
hello welcome backsuresh
```

Removing sessions:

- We can remove session variables and completely destroy sessions. To do this we need to use methods.

```
Unset ($_SESSION['name']);
```

- It simply unset the corresponding key of the SESSION array.
- If you want to destroy session completely, you need to use session_destroy method.

```
session_destroy();
```

- It is necessary to call session_start() again after the session_destroy() method if you want new session again to start.

Additional Session Functions:

- Apart from this there are other session methods available to work with sessions, we will list them.
  1. **session_abort():** This function discards the session array changes and finish session.

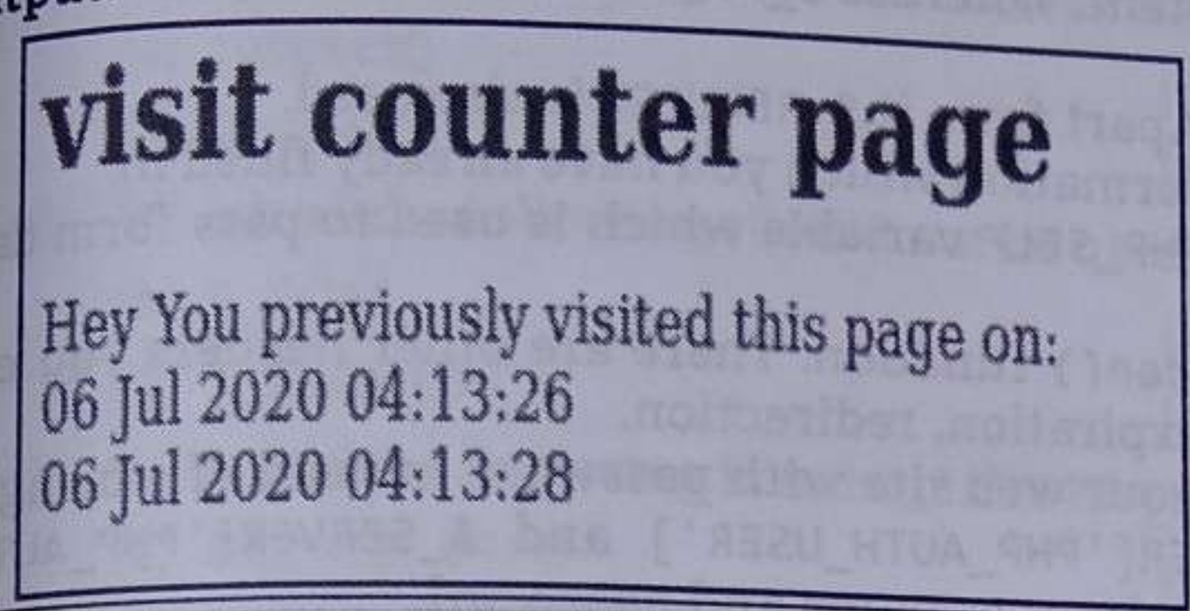
     Syntax: `session_abort(void): bool`
  2. **session_unset():** It removes all the variables assigned to $_SESSION.
  3. **session_generate_id():** It creates a new session id.
  4. **session_name([$string]):** It returns the name of the session. If `$string` is given, the session name will be set to value of string.
  5. **session_id():** It returns a string containing session id.

     Example: echo session_id();
  6. **session_regenerate_id():** It updates current session id with newly generated id.
  7. **session_write_close():** This method is used to end the current session and store current session data. This method will be automatically called at the end of page.
- We will create a small application to see how many number of time client visits to a page using session method.

Program 4.21: PHP program for user visit count using session.

```
<?php
// it start session
session_start();
?>
<?php
echo "<h1> visit counter page</h1>";
if (!isset($_SESSION['visit_count']))
{
   echo 'Hello its your first visit.';
}
else
{
   echo 'Hey You previously visited this page on: <br/>';
   foreach ($_SESSION['visit_count'] as $val)
   {
         echo date('d M Y h:i:s', $val) . '<br/>';
   }
}
?>
<?php
$_SESSION['visit_count'][] = mktime();
?>
```

Output:



4.7 SSL

- In this age of information insecurity, all websites that handle sensitive customer information such as passwords, credit card numbers etc., must offer secure and encrypted communication channels.
- In the absence of such channels, intercepting and stealing customers' information becomes quite easy for anyone.
- A Secure Socket Layer (SSL) certificate encrypts the connection between the customer and your web server so that the information is rendered unreadable by all third parties (identity thieves and hackers).
- The Secure Socket Layer (SSL) provides secure channel over which regular HTTP requests and response can flow. You can not control encryption from PHP. `https://` URL indicates secure connection for that document unlike `http://` URL.
- The HTTPS entry in the `$_SERVER` array is set to on if the PHP page was entered in response to a request over an SSL connection. To prevent a page from being generated over a non-encrypted connection, you should use,

```
if($_SERVER['HTTPS']=='on')
{
    die('it must be a secure connection');
}
```

- You can add SSL certificate to site with following steps:
  (1) Set up SSL certificate signing request.
  (2) Create SSL certificate.
  (3) Install SSL on PHP on cloudways.
  (4) Testing HTTPS.

Summary

- In this chapter, we have learnt `$_SERVER` is an auto global array which contains number of useful elements that provide information on to the web server and current request.
- `$_SERVER` array contains useful information from the web server. `$_SERVER` is a super global variable it contains information about headers, path, host and script locations. It has different parameters like `PHP_SELF`, `SERVER_INFO`, `SERVER_NAME`, `SERVER_PORT`, `SERVER_ID` etc.
- It is very easy to process forms with php, you can process information collected by HTML form and you can use PHP code to make decisions based on this information to create dynamic web pages.

- $_GET and $_POST are used to process form data. $_GET is not so secure and it is used to send very less data also it is idempotent. Whereas $_POST is secured and you can send huge amount of data with it.
- Generally $_POST is recommended. Apart from it $_REQUEST is also used.
- Sticky form is used to store form information which you have already filled in.
- Self processing page is used with $PHP_SELF variable which is used to pass form data to same page.
- Header information is set with header() function. There are other headers you can use to set header like Content-type, expiration, redirection.
- If you want PHP to protect parts of your web site with password instead of storing in external file you can use $_SERVER['PHP_AUTH_USER'] and $_SERVER['PHP_AUTH_PASSWORD'] global variables that contains username and password.
- To keep track of state information between requests and to maintain state cookies are used.
- A cookie is a bit of information that server can give to client. On every subsequent request the client will give information back to server, thus identifying itself.
- Cookies are useful for retaining information through repeated visits by browser, but many browsers allow users to disable cookies. So the best way to maintain state with PHP is to use built-in session tracking mechanism.
- A PHP session is easily started by making a call to the session_start() function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page.
- Session variables are stored in associative array called $_SESSION[]. These variables can be accessed during lifetime of a session.
- A Secure Socket Layer (SSL) certificate encrypts the connection between the customer and your web server so that the information is rendered unreadable by all third parties.
- The Secure Socket Layer (SSL) provides secure channel over which regular HTTP requests and response can flow. You can not control encryption from PHP.

Check Your Understanding

Q.I Multiple choice questions.

1. Which of the following PHP variables are used to process form data?
   (a) $_GET & $_PUT (b) $_GET & $_POST
   (c) $_SET & $_GET (d) READ & WRITE
2. When $_GET variable is used to collect form data, the data is visible _____.
   (a) to nobody (b) to only one user
   (c) to root user (d) to everyone
3. How many predefined variables are used in PHP to authenticate use?
   (a) 3 (b) 4
   (c) 2 (d) 1
4. Which of the following PHP variables are uses for authentication?
   (i) $_SERVER['PHP_AUTH_USER];
   (ii) $_SERVER['PHP_AUTH_ROOT'];
   (iii) $_SERVER['PHP_AUTH_PWD'];
   (iv) $_SERVER['PHP_AUTH_PW'];
   (a) (i) and (ii) (b) (i) and (iii)
   (c) (i) and (iv) (d) (ii) and (iii)

5. Which of the following PHP function is used for authentication?
   (a) inspect()
   (b) intersect()
   (c) header()
   (d) footer()
6. Which of the following should not be used while sending crucial information?
   (a) $_POST
   (b) $_GET
   (c) $_PROCESS
   (d) $_ISSET
7. Which of the following is not $_SERVER parameter?
   (a) SERVER_INFO
   (b) PHP_SERVER
   (c) SERVER_NAME
   (d) SERVER_PORT
8. Which is true about $PHP_SELF?
   (a) It passes PHP form information to next page.
   (b) It stores web page information.
   (c) It is used to debug page.
   (d) It is used to pass page information to itself.
9. What is correct statement about sticky forms?
   (a) It is simply HTML form that remembers data how you filled it.
   (b) It is used to submit data to another page.
   (c) It is used to pass data to server.
   (d) None of above.
10. What is use of $isset() function?
   (a) It is used to check whether variable is set or not.
   (b) It is used to set variable.
   (c) It is used to set new value.
   (d) All of above.

ANSWERS

1. (b)	2. (d)	3. (c)	4. (c)	5. (c)	6. (b)	7. (b)	8. (d)
9. (a)	10. (a)						

Q.II State True/False.

(1) PHP $_GET[] is superglobal array in PHP.
(2) $is_set() is super global array in PHP.
(3) sticky form is used to store customer data.
(4) Cookie is stored at server side.
(5) Session is stored at client side.
(6) Input is form element.
(7) Redirection is used for setting response headers.

ANSWERS

1. True	2. False	3. True	4. False	5. False
6. True	7. True			

Practice Questions

Q. I Answer the following questions in short.

1. What is $_SERVER variable ?
2. List different HTML form elements.
3. What is use of isset() method ?
4. Why we use $_REQUEST[]?
5. Describe following headers:
   (i) Expiration
   (ii) Redirection
   (iii) Location
   (iv) Content-type
6. What is SSL?

Q. II Answer the following questions.

1. Explain different parameters used for $_SERVER information?
2. What is sticky form? Explain.
3. How to set response headers? Justify?
4. How to perform Form processing PHP? Explain.
5. What is self processing form? Explain.
6. Differentiate between $_GET and $_POST variables.
7. Explain what is HTTP authentication?
8. Explain how to maintain state in PHP?
9. Write a PHP Script to create Session?
10. Write a PHP script to destroy cookie which is created.
11. Write a PHP script to demonstrate SSL

Q. III Define the terms

1. Forms
2. Self processing page
3. Sticky form
4. HTML element
5. GET method
6. POST method
7. cookie
8. Session

***

5... Databases

Objectives...

- To know about the basics of interaction with database in PHP.
- To get familiar with the different methods used to interact with the database.
- To develop basic programs of database interaction.

5.1 INTRODUCTION

- A database acts as a backend. Database stores a large amount of related data. A collection of related tables forms a database. Each row in a table is called a record and each column in a record is called a field. Using PHP, we can add records, retrieve records, update records of the table in the database. PHP supports databases like Ingres, Oracle, dBase, InterBase, PostgreSQL, MySQL, Sybase, IBM DB2 etc.

5.2 USING PHP TO ACCESS A DATABASE

- There are two methods to access databases from PHP. These are :

  (i) Database Specific Method,

  (ii) Database Independent Method (PEAR DB).

Steps to open and interact with a database through PHP:

1. Open a connection to the database by passing a connection string (two methods have different syntax).

   In case of database specific method, we use `pg_connect()` function for connecting to PostgreSQL database. In case of database independent method (PEAR DB), we use `DB::connect()`.

   For example,

   (i) Database Specific Method:

```
$db=pg_connect("host=localhost  port=5432  dbname=sybca  user=postgres
password=root") or die("Cannot Connect");
```

   (ii) Database Independent Method:

```
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
```

2. Execute the query on the connection created in first step. For database specific method, `pg_query()` is used. For database independent method, `query()` is used.
3. Process the result set obtained after executing the query. For retrieval queries, we need to process the result. This is because the query may return more than one record in the result set. For both the methods, there are various functions.

   For example, `pg_fetch_row()` is used for database specific method. `fetchRow()` is used for database independent method.
4. Release the resources. This means free the result variable and free the connection resource. For database specific method, `pg_free_result()` and `pg_close()` functions are used. For database independent method, `free()` and `disconnect()` functions are used.

5.2.1 Database Specific Method

- As there is a database specific prefix before the function, it is easy to identify the database specific method.

5.2.1.1 Opening a Database Connection

- There are database specific prefixes to connect to a specific database. For example, if we want to connect to PostgreSQL database we use `pg_connect()` function. If we want to connect to MySQL database we use `mysql_connect()` function.
- **Syntax:**

```
resource pg_connect( string $conn_string )
```

- The `$conn_string` can have one or more parameter setting separated by white space. The list of parameters is : host, port, dbname, user, password. On success, the `pg_connect()` function returns PostgreSQL connection resource. On failure, it returns false.
- For example,

```
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root")
or die("Cannot connect");
```

- In the above example, we connect to a database named "sybca".

```
$db=pg_connect("dbname=sybca user=postgres password=root")
or die("Cannot connect");
```

- In the above example, we connect to a database named "sybca". Here, the default values are taken for the remaining parameters.

5.2.1.2 Closing a Database Connection

- `pg_close()` function is used to close a PostgreSQL connection.
- **Syntax:**

```
bool pg_close( [ resource $conn_resource ] );
```

- The $conn_resource is a PostgreSQL database connection resource. Generally, pg_close() is not required as non-persistent connections are closed at the end of the script automatically.
- The pg_close() function returns true on success or false on failure.

Program 5.1: Write a PHP program to check whether a connection to a database is successful or not.

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root") or die("Cannot connect");
echo "Connection is successfully done";
pg_close($db);
?>
```

Output: Connection is successfully done

5.2.1.3 Executing a Query on a Database Connection

- pg_query() function executes a query on the specified database connection.
- **Syntax:**

```
resource pg_query( [ resource $conn_resource ], string $query_string );
```

- If the first parameter is omitted, the default connection is used. But it is not recommended to keep the first parameter option, as it can lead to the problems which can be hard to debug. The function returns query result resource on success or false on failure.
- For example,

```
$result=pg_query($db,"SELECT * FROM stud") or die("Cannot execute");
```

5.2.1.4 Processing the result set

1. **pg_fetch_row:**

- The pg_fetch_row() function returns the values of all the columns in a record.
- **Syntax :**

```
array pg_fetch_row( resource $result [, integer $row ] )
```

- The fields are indexed with number starting with zero(0). The second parameter is optional.
- If pg_fetch_row() is called with only first parameter, it function returns the next row or false when no more rows remain in the result set. The second parameter is the row number in the result to fetch. The rows are numbered in the increasing order starting from zero(0).

Program 5.2: Write a PHP program to display students studying in 'fybca' class using pg_fetch_row() function.

```
<?php
$db=pg_connect("host=localhost  port=5432  dbname=sybca  user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db,"SELECT * FROM stud") or die("Cannot execute");
while($row=pg_fetch_row($result))
{
        echo "Roll No. : $row[0]  Name :  $row[1]";
}
pg_free_result($result);
pg_close($db);
?>
```

Output:

Roll No. : 101 Name : Mahesh
Roll No. : 102 Name : Anuradha
Roll No. : 103 Name : Akash
Roll No. : 104 Name : Sagar
Roll No. : 105 Name : Pranav

2. pg_fetch_array:

- The `pg_fetch_array()` function returns the row as an array. It returns an array containing each field value for the given row.
- **Syntax:**

  `array pg_fetch_array( resource $result [, integer $row, integer $result_type ] )`
- Every call to `pg_fetch_array()` function returns the next row, or false when there are no more rows are remaining. The optional parameter `$result_type` gives you option to specify the type of returned array.
- It takes the following values: `PGSQL_NUM`, `PGSQL_ASSOC`, `PGSQL_BOTH`. If we use `PGSQL_NUM`, it returns an indexed array with numerical indices.
- If we use `PGSQL_ASSOC`, it returns an array with associative array with indices having field names. If we use `PGSQL_BOTH`, it returns both numerical and associative indices.

Program 5.3: Write a PHP program to display the students studying in 'sybca' class using pg_fetch_array() function.

```
<?php
$db=pg_connect("host=localhost  port=5432  dbname=sybca  user=postgres
password=root") or die("Cannot connect");
```

```
$result=pg_query($db,"SELECT * FROM stud WHERE class_name='sybca'") or
die("Cannot execute");
while($row=pg_fetch_array($result))
{
    echo "Roll No. : $row[0]  Name :  $row[1]";
}
pg_free_result($result);
pg_close($db);
?>
```

Output:

Roll No. : 4 Name : pqr

Roll No. : 5 Name : jkl

---

3. pg_fetch_assoc:

- The `pg_fetch_assoc()` function returns the row as an associative array. It is equivalent to `pg_fetch_array()` with `PGSQL_ASSOC` as the optional third parameter.
- **Syntax:**

```
array pg_fetch_assoc( resource $result [, int $row ] )
```

Program 5.4: Write a PHP program to display the students studying in 'sybca' class using pg_fetch_assoc() function.

```
<?php
$db=pg_connect("host=localhost    port=5432    dbname=sybca    user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db,"SELECT roll_no,name FROM stud WHERE
class_name='sybca'") or die("Cannot execute");
while($row=pg_fetch_assoc($result))
{
    echo "Roll No. : $row['roll_no']  Name :  $row['name']";
}
pg_free_result($result);
pg_close($db);
?>
```

Output:

Roll No. : 4 Name : pqr

Roll No. : 5 Name : jkl

4. pg_fetch_object:

- The `pg_fetch_object()` function fetches the row as an object.
- **Syntax:**

```
object pg_fetch_object( resource $result [, integer $row])
```

- `pg_fetch_object()` returns an object with properties that are nothing but fetched row's field names. The `$row` is the row number in the result to fetch. The rows are numbered in increasing order starting from zero(0). If called with single parameter each call to `pg_fetch_object()` function returns the next row or false when no more rows remain in the result set.

Program 5.5: Write a PHP program to display student information of students studying in 'sybca' class using pg_fetch_object() function.

```
<?php
  $db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
  password=root") or die("Cannot connect");
  $result=pg_query($db,"SELECT roll_no,name FROM stud WHERE class='sybca'")
  or die("Cannot execute");
  while($row=pg_fetch_object($result))
  {
     echo "Roll No. : $row->roll_no  Name :  $row->name";
  }
  pg_free_result($result);
  pg_close($db);
?>
```

Output:

Roll No. : 4 Name : pqr

Roll No. : 5 Name : jkl

5. pg_fetch_result:

- The `pg_fetch_result()` function returns the value of a user specified row and column in a result set.
- **Syntax:**

```
string pg_fetch_result( resource $result, integer $row, mixed $column )
string pg_fetch_result( resource $result, mixed $column )
```

- `$row` is the row number in the result to fetch. The rows are numbered from zero(0) onwards. If `$row` parameter is not mentioned, next row is fetched. `$column` is either a string representing the name of the column to fetch or an integer describing which column number to fetch. Columns are numbered in increasing order starting from zero(0).

Program 5.6: Write a PHP program to display the class name of first student and value in the second column of the first student studying in 'sybca'.

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=Postgres
    password=root") or die("Cannot connect");
$result=pg_query($db, "SELECT roll_no,name,class FROM stud WHERE
  class='sybca'") or die("Cannot execute");
//Display the "class" of "first" student
print(pg_fetch_result($result, 0, 2));
echo "<br>";
//Display the value in the "second" column (name) of the first student
print(pg_fetch_result($result, 1));
pg_free_result($result);
pg_close($db);
?>
```

Output :

```
sybca
pqr
```

6. **pg_fetch_all:**

- The `pg_fetch_all()` function returns all rows from a result in the form of an array. PHP assigns each row with an integer starting with zero(0). Each element is itself an array in which the column names are used as indices.
- **Syntax:**

```
array pg_fetch_all( resource $result )
```

Program 5.7: Write a PHP program to display the student details who are studying in 'sybca' using pg_fetch_all() function.

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db,"SELECT roll_no,name,class FROM stud WHERE
class='sybca'") or die("Cannot execute");
print_r(pg_fetch_all($result));
pg_free_result($result);
pg_close($db);
?>
```

Output:

```
Array([0] => Array([roll_no] => 4 [name] => pqr [class] => sybca [1] =>
Array([roll_no] => 5 [name] => jkl [class] => sybca))
```

7. pg_num_fields:

- This function returns the total number of columns in a result set. On error it returns –1.
- **Syntax:**

```
integer pg_num_fields( resource $result )
```

Program 5.8: PHP program to display the total number of columns returned by a query containing the students studying in 'sybca' class.

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db, "SELECT roll_no,name,class FROM stud WHERE
class='sybca'") or die("Cannot execute");
print("Total number of columns : ".pg_num_fields($result));
pg_free_result($result);
pg_close($db);
?>
```

Output : Total number of columns : 3

8. pg_num_rows:

- This function returns the total number of rows in a result set. On error, it returns –1.
- **Syntax:**

```
integer pg_num_rows( resource $result )
```

Program 5.9: PHP program to display the total number of rows returned by a query containing students studying in 'sybca' class.

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db, "SELECT roll_no,name,class FROM stud WHERE
class='sybca'") or die("Cannot execute");
print("Total number of rows : "pg_num_rows($result));
pg_free_result($result);
pg_close($db);
?>
```

Output : Total number of rows : 6

9. pg_field_name:

- This functionis used to returns the field(column) name.

- **Syntax:**

  `string pg_field_name( resource $result, integer $column_number )`

- Field(column) numbering starts from zero(0). It returns field(column) name on success and false on error.

10. pg_field_num:

- This function returns the field(column) number of the named field(column).
- **Syntax:**

  `integer pg_field_num( resource $result, string $column_name )`

- It returns the field(column) number or -1 on error. The field(column) numbering start from zero.
- Both the `pg_field_name()` and `pg_field_num()` functions are used in the following example.

Program 5.10: PHP program to display the first column name and the column number of field 'class' in the result of a query containing students of 'sybca' class.

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db, "SELECT roll_no,name,class FROM stud WHERE
class='sybca'") or die("Cannot execute");
print("First Column is : ".pg_field_name($result,0));
echo "<br>";
print("The column number of class is : ".pg_field_num($result,"class"));
pg_free_result($result);
pg_close($db);
?>
```

Output :

```
First Column is : roll_no
The column number of class is : 2
```

11. pg_result_error:

- This function displays the error message related with the result.
- **Syntax:**

  `string pg_result_error( resource $result )`

- In the following, we try to retrieve from a nonexistent "student" table. It displays an error message.

Program 5.11: PHP program to demonstrate the use of pg_result_error() function and display the error message(if any).

```
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root");
if(!pg_connection_busy($db))
{
    pg_send_query($db, "SELECT * FROM student;");
}
$result=pg_get_result($db);
echo pg_result_error($result);
?>
```

Output :

ERROR: relation "student" does not exist LINE 1: SELECT * FROM student;^

- The pg_send_query() function sends asynchronous query to the database. The pg_get_result() gets asynchronous query result. The pg_connection_busy() function checks whether the connection is busy or not. It waits if the previous query is still running, in that case pg_get_result() remains blocked.

5.2.1.5 Releasing the Result Memory

- The pg_free_result() function is used to free the memory occupied by the PostgreSQL query result.
- **Syntax:**

```
boolean pg_free_result( resource $result )
```

5.3 PEAR DB BASICS (DATABASE INDEPENDENT METHOD)

- The PHP Extension and Application Repository (PEAR) is a framework.
- PEAR DB is an advanced, object-oriented database library providing us the full database abstraction.
- It is also known as the database independent method. It means that the same code is applicable to all databases which are not possible with the database specific method.

5.3.1 Opening a Database Connection

5.3.1.1 Data Source Names

- In order open a database connection using PEAR DB, first we need to build the DSN. A data source name (DSN) is a string having different components like type of database, username, password, protocol, host specification, database name.

- **The DSN has the following Syntax:**

```
dbsyntax://username:password@protocol+hostspec/database
```

- Following table specifies the currently supported database types.

Table 5.1: Supported database types by PHP

dbsyntax	Database
mysql	MySQL
pgsql	PostgreSQL
msql	Mini SQL
ibase	InterBase
Sybase	Sybase
fbsql	FontBase
ifx	Informix
mssql	Microsoft SQL Server
oci8	Oracle 7/8/9
odbc	ODBC(Open Database Connectivity)

- The protocol is the communication protocol to use. The common values are "tcp" and "unix" which are related with Internet and Unix domain sockets. Every database does not necessarily support every communication protocol. Following are some valid data source names.

```
pgsql://postgres:root@tcp+localhost/sybca
pgsql://postgres:root@localhost/sybca
pgsql://postgres@localhost/sybca        //if there is no password
```

5.3.1.2 Opening a Connection

- After building a DSN, we can open a connection using the `connect()` method. We first need to include `'DB.php'` header file in our code by using function `require_once('DB.php');`
- **The syntax for connecting to a database:**

```
$db=DB::connect(DSN [, options]);
```

- The options part is not mandatory. The options can be 'persistent', 'optimize' or 'debug'. The connection is not persistent and no debugging information is display by default. The permissible values for 'optimize' option are 'portability' and 'performance' (default).
- For example,

```
$dsn=" pgsql://postgres:root@localhost/sybca";
$db=DB::connect($dsn,array('debug'=>1, 'optimize'=>'performance'));
```

5.3.2 Checking for the Errors

- By using `DB::iserror()` method, we can check whether there is an error or not.
- **Syntax:**

```
$db=DB::connect($dsn);
if(DB::iserror($db))
{
    die($db->getMessage( ));
}
```

- For example,

```
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
    die($db->getMessage( ));
}
```

- In the above example, `$db` is the database connection resource. The above code checks whether any error occurs while the connection is established with the database.
- The `DB::iserror()` returns true if there is an error occurred. In that case, the program execution is stopped using `die()` method and an error message is displayed by the `getMessage()` method.

5.3.3 Executing a Query

- The `query()` method executes a query on the database. This method dispatches SQL query to the database.
- **Syntax:**

```
$result=$db->query($sql);
```

- Where `$sql` is the SQL statement passed as a parameter and `$db` is the database connection resource.
- In case of DML queries having INSERT, DELETE, UPDATE commands, the method returns the DB_OK constant to indicate success. In case of SELECT command, the method returns an object that can be used to access the result.
- To check whether the query is successfully completed or not `DB::iserror()` is used as follows:
- **Syntax:**

```
$result=$db->query($sql);
if(DB::iserror($result))
{
    die($result->getMessage());
}
```

5.3.4 Fetching Results from a Query

- PEAR DB has two methods for retrieving data from a query result set. The `fetchRow()` method returns an array while `fetchInto()` method stores it into an array.

5.3.4.1 Returning the Row

- The `fetchRow()` method returns an array of the next row of results.
- **Syntax:**

```
$row=$result->fetchRow([ mode ]);
```

- This method returns either an array of data (if there is some data), NULL (if there is no more data), or DB_ERROR (if an error occurred). The mode parameter is optional. If it is mentioned, specifies the format of the array returned.
- Generally, the `fetchRow()` method is used in a while loop to process a result, one row at a time.
- **Syntax:**

```
while($row=$result->fetchRow())
{
    if(DB::iserror($row))
    {
        die($row->getMessage());
    }
//statements to process the row
}
```

5.3.4.2 Storing the Row

- The `fetchInto()` method instead of returning the next row, stores it into an array variable passed as a parameter.
- **Syntax:**

```
$answer=$result->fetchInto(array, [,mode]);
```

- Like `fetchRow()`, `fetchInto()` returns NULL (if there is no more data), or DB_ERROR (if an error occurred). To process the result one row at a time `fetchInto()` is used as follows:

```
while($answer=$result->fetchInto($row))
{
    if(DB::iserror($answer))
    {
        die($answer->getMessage());
        //statements to process the row
    }
}
```

- The optional mode parameter to `fetchRow()` or `fetchInto()` controls the format of the row array.
- The possible values are DB_FETCHMODE_ORDERED(default) and DB_FETCHMODE_ASSOC, DB_FETCHMODE_OBJECT.
- The DB_FETCHMODE_ASSOC creates an array whose keys are column names and whose values are the values from those columns.
- The DB_FETCHMODE_ORDERED creates an indexed array. To access the data in the object format, use DB_FETCHMODE_OBJECT mode.

For Example:

1. **Using default mode, i.e. DB_FETCHMODE_ORDERED(no need to mention)**

```
$result=$db->query("SELECT * FROM stud");
if(DB::iserror($result))
{
    die($result->getMessage());
}
while($row=$result->fetchRow())
{
   echo $row[0]." ".$row[1]." ".$row[2]."<br>";
}
```

2. **Using DB_FETCHMODE_ASSOC.**

```
$result=$db->query("SELECT * FROM stud");
if(DB::iserror($result))
{
   die($result->getMessage());
}
while($row=$result->fetchRow(DB_FETCHMODE_ASSOC))
{
   echo $row['roll_no']." ".$row['name']." ".$row['class']."<br>";
}
```

3. **Using DB_FETCHMODE_OBJECT.**

```
$result=$db->query("SELECT * FROM stud");
if(DB::iserror($result))
{
   die($result->getMessage());
}
while($row=$result->fetchRow(DB_FETCHMODE_OBJECT))
{
   echo $row->roll_no." ".$row->name." ".$row->class."<br>";
}
```

5.3.4.3 Finishing the Result

- A query result object holds all the rows fetched by the query. This may require a lot of memory for storing the result object. When the processing on the result object is over, the memory occupied by it is released using the `free()` method.
- **Syntax:**

```
$result->free();
```

- This is not mandatory as `free()` is called automatically when the PHP script ends.

5.3.5 Disconnecting

- To release the database connection, use the `disconnect()` method on the database object.
- **Syntax:**

```
$db->disconnect();
```

- This is not mandatory, as all the database connections are terminated when the PHP script ends.

5.4 ADVANCED DATABASE TECHNIQUES

- In addition to the basic database methods, PEAR DB provides several other methods for fetching rows, likewise a unique row ID system and a pair of prepare/execute statements that can improve the performance of repeated queries.

5.4.1 Placeholders

- While writing the query like INSERT or UPDATE, we can mention a placeholder '?' in place of specific value. In the `query()` method, we can add a second parameter consisting of an array of values to insert in place of '?' into the SQL.
- **Syntax:**

```
$result=$db->query(SQL, values);
```

- For example, the following code inserts two entries into the stud table.

```
$studinfo=array(array(10,'Mandar','sybca'),array(11,'Chinmay','fybca'));
foreach($studinfo as $s)
{
    $db->query('INSERT INTO stud(roll_no,name,class) VALUES(?,?,?)', $s);
}
```

- Three characters that can be used as placeholder values in an SQL query are as follows:
  - **?** : A string or number, which will be quoted if necessary (recommended).
  - **|** : A string or number, which will never be quoted.
  - **&** : A filename, contents of which will be included in the statement.

5.4.2 Prepare/Execute and Prepare/Execute Multiple

- If we want to execute the same query repeatedly, then it is always better to compile the query once and then execute the compiled query many times. This can be done by using the `prepare()`, `execute()` and `executeMultiple()` methods.
- The `prepare()`, `execute()` and `executeMultiple()` methods are used in a pair, which means either you have to use `prepare()` and `execute()` or you have to use `prepare()` and `executeMultiple()`.
- The use of `prepare()` and `execute()` is explained below.

Step 1 : Call prepare() method on the query.

```
$compiled_query=$db->prepare($sql);
```

`$sql` is the SQL statement. The `prepare()` method returns the compiled version of the query object.

Step 2 : The `execute()` method fills in any placeholders in the query by using second parameter and sends it back to RDBMS.

```
$response=$db->execute($compiled_query, $values);
```

- In case of an error, DB_ERROR is returned, otherwise query response object is returned. The $values array contains the values for the fields(placeholders) mentioned in the SQL query. The placeholders are shown with '?' in the SQL query.
- For example, in the following example two student records are added to the stud table after the successful execution of `prepare()` and `execute()` methods.

```
<?php
//Using prepare() and execute() methods
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$st=array(array(7,'Anil','tybca'), array(9,'Sanjiv','sybca'));
$sql="INSERT INTO stud(roll_no, name, class) VALUES(?,?,?)";
$compiled_query=$db->prepare($sql);
foreach($st as $val)
{
   $db->execute($compiled_query,$val);
}
$db->disconnect();
?>
```

- The use of `prepare()` and `executeMultiple()` methods is explained below.

Step 1 : Call prepare() method on the query.

```
$compiled_query=$db->prepare($sql);
```

`$sql` is the SQL statement. The `prepare()` method returns the compiled version of the query object.

Step 2 : The executeMultiple() method avoids the iterations of the foreach loop mentioned for the execute() method.

Syntax of executeMultiple:

```
$response=$db->executeMultiple($compiled_query, $values);
```

The `$values` array is two dimensional indexed array.

For example, in the following example two student records are inserted in stud table.

```
<?php
//Using prepare() and executeMultiple() methods
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$st=array(array(7,'Anil','tybca'), array(9,'Sanjiv','sybca'));
$sql="INSERT INTO stud(roll_no, name, class) VALUES(?,?,?)";
$compiled_query=$db->prepare($sql);
//Iterations of foreach loop is not required.
$db->executeMultiple($compiled_query,$st);
$db->disconnect();
?>
```

5.4.3 Other Methods

- PEAR DB provides other methods that execute a query and fetch the results in single step. These methods are `getOne()`, `getCol()`, `getRow()`, `getAssoc()`, and `getAll()`. All of these methods support placeholders.

1. **getOne():**

- This method executes a query and returns the first column of the first row.
- **Syntax:**

```
mixed getOne(string $sql [, mixed $param])
```

Here `$sql` is a SQL query and `$param` can numeric, array or string.

Program 5.12: PHP program to display the first column of the first row of the result containing students of 'fybca' class.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
   $sql="SELECT name, class FROM stud WHERE class='fybca'";
$row=$db->getOne($sql);
print_r($row);
$db->disconnect();
?>
```

Output: Mahesh

2. getCol():

- The getCol() method returns a single column from the result returned by the SQL query.
- **Syntax:**

```
array getCol( string $sql [, mixed $col, mixed $values ] )
```

- Here, $sql is the SQL query, $col can be integer or string specifying which column to return, $values can be number, string or array which specify parameters for SELECT statement (if any). Last two parameters are optional.

Program 5.13: PHP program to display the value in the specified column 'class' of a student having roll number 105.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$sql="SELECT name, class FROM stud WHERE roll_no=105";
$row=$db->getCol($sql,"class");
print_r($row);
$db->disconnect();
?>
```

Output: Array([0]=>fybca)

3. **getRow():**

- The getRow() method returns the first row of result data returned by the SQL query.
- **Syntax:**

```
array getRow(string $sql [, array $param ])
```

Here `$sql` is the SQL query, `$param` is array of parameters.

Program 5.14: PHP program to display the name and class of a student whose roll number is 9 using getRow() method (pass roll number as a parameter).

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$sql="SELECT name, class FROM stud WHERE roll_no=?";
$a=array(9);
$row=$db->getRow($sql,$a);
print_r($row);
$db->disconnect();
?>
```

Output: `Array([0]=>Sanjiv [1]=>sybca )`

- In the above example, the name and class of student whose roll number is 9 is returned as an array.

4. **getAssoc():**

- The getAssoc() method returns the whole result set in the form of an associative array.
- **Syntax:**

```
array getAssoc( string $sq [, boolean $opt])
```

Here `$sql` is the SQL query and `$opt` is used when query returns exact two fields.

Program 5.15: PHP program to display name and class of a student whose roll number is 9 using getAssoc method.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
```

```
{
    die($db->getMessage());
}
$sql="SELECT name, class FROM stud WHERE roll_no=9";
$row=$db->getAssoc($sql,true);
print_r($row);
$db->disconnect();
?>
```

Output: `Array ( [Sanjiv] => Array ( [0] => sybca ) )`

5. getAll()

- The `getAll()` method returns an array containing all the rows returned by the SQL query. If we specify the placeholder in the SQL statement, it retrieves only specific rows.
- **Syntax:**

  `array getAll(string $sql [, array $param [, mixed $mode]])`
- Here, `$sql` is the SQL query, `$param` is an array of parameters and `$mode` is either numeric or in string format.
- In the following example, `roll_no` is passed as a parameter to the SQL query. As a result of this, we found that the name and class of the specified `roll_no` is displayed.

Program 5.16: PHP program to display the name and class of a student whose roll number is 9 using getAll. (pass roll number as a parameter to getAll).

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
    die($db->getMessage());
}
$sql="SELECT name, class FROM stud WHERE roll_no=?";
$a=array(9);
$row=$db->getAll($sql,$a,2);      //2 is equivalent with DB_FETCHMODE_ASSOC
print_r($row);
$db->disconnect();
?>
```

Output: `Array( [0]=> Array ( [name]=>Sanjiv [class]=>sybca ) )`

5.4.4 Getting information about Query Response

- PHP provides four PEAR DB methods to get information about query result object. These are `numRows()`, `numCols()`, `affectedRows()` and `tableInfo()`.

1. **numRows():**

- The `numRows()` method returns the total number of rows returned by the SELECT query.
- **Syntax:**

```
integer numRows()
```

2. **numCols():**

- The `numCols()` method returns the total number of columns returned by the SELECT query.
- **Syntax:**

```
integer numCols()
```

- In the following example, both the numRows() and numCols() methods are used.

Program 5.17: PHP program to display the total number of rows and columns returned by a query result.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$sql="SELECT name, class FROM stud";
$result=$db->query($sql);
echo "Total number of rows returned is: ".$result->numRows();
echo "<br>";
echo "Total number of columns returned is:".$result->numCols();
$result->free();
$db->disconnect();
?>
```

Output: Total number of rows returned is:5
Total number of columns returned is:2

3. **affectedRows():**

- The INSERT, UPDATE, DELETE statements affect the number of rows in a table.

- If we use INSERT statement, then the number of rows will increase. In the same way, if we use either UPDATE or DELETE statements with specific condition, then some specific number of records gets affected.
- The `affectedRows()` method gives us the information about the number of rows affected by UPDATE, INSERT or DELETE statements.
- **Syntax:**

```
integer affectedRows()
```

Program 5.18: PHP program to update the class to 'sybca' of students whose roll number is 4 and 5. Also display the total number of affected rows by this update command.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
    die($db->getMessage());
}
$sql="UPDATE stud SET class='sybca' WHERE roll_no IN(4,5)";
$db->query($sql);
echo "Affected rows: ".$db->affectedRows();
$db->disconnect();
?>
```

Output: Affected rows: 2

5.4.4.1 tableInfo()

- The `tableInfo()` method returns the detail information on the data type and other points like flags of columns returned from a SELECT operation.
- **Syntax:**

```
array tableInfo(mixed $result )
```

- The following code finds the information about a table 'stud'.

Program 5.19: PHP program to display the detail information about 'stud' table.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
    die($db->getMessage());
```

```
}
$info=$db->tableInfo('stud');
echo "<table border='2'>";
foreach($info as $key=>$val)
{
    echo "<tr>";
    foreach($val as $k=>$v)
    {
        echo "<tr><td>$k</td><td>$v</td></tr>";
    }
    echo "</tr>";
}
echo "</table>";
$db->disconnect();
?>
```

Output :

table	stud
name	roll_no
type	numeric
len	-1
flags	not_null primary_key
table	stud
name	name
type	text
len	-1
flags	not_null
table	stud
name	class
type	text
len	-1
flags	not_null

5.4.5 Metadata

- The `getListOf()` method query the database and gives information on available databases, users, views, and functions.
- **Syntax:**

  `array getListOf(string $param)`

  Here `$param` can be tables, databases, users, functions, views etc.
- The following script displays all the views created in "sybca" database.

Program 5.20: PHP program to display all the views defined in 'sybca' database.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$data=$db->getListOf('views');
print_r($data);
$db->disconnect();
?>
```

Output: Array ([0] => emp_view)

5.4.6 Transactions

- A transaction is an atomic unit of execution which is performed as a whole unit or none of the instruction from the transaction is executed.
- A series of changes made to the database can be committed (all changes applied) or rolled back(changes not applied).
- For example, in a transaction of transferring an amount from account of person 'A' to account of person 'B', the withdrawal from A's account and deposit into B's account must happen as a whole unit(together). These two actions cannot be performed independently as a separate instruction. If it happens, it leaves the database in inconsistent state, which is not acceptable. PEAR DB offers `commit()` and `rollback()` methods to help with transactions.

  ```
  $result = $db->commit();
  $result = $db->rollback();
  ```
- Calling either `commit()` or `rollback()` on a non supported database returns DB_ERROR.

5.4.7 Sequences

- PEAR DB sequences are somewhat similar to the database specific ID assignment. The `nextId()` method returns the next ID for that given sequence.
- **Syntax:**

```
integer nextId( string $sequence_name )
```

- A sequence is a table in the database that maintains the last-assigned ID. We can create and delete sequences with `createSequence()` and `dropSequence()` methods.
- **Syntax:**

```
integer createSequence( string $sequence_name )
integer dropSequence( string $sequence_name )
```

- The following example creates a sequence named `'newseq'`.

Program 5.21: PHP program to create a new sequence named 'newseq' and display the next ID for that sequence.

```
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
   die($db->getMessage());
}
$ret=$db->createSequence('newseq');
var_dump($ret);
$id=$db->nextId('newseq');
echo "<br>The next id is : ".$id;
$db->disconnect();
?>
```

Output:

```
int(1)
The next id is : 1.
```

PROGRAMS

Program 5.22: Write a PHP program which accepts the department name from user and displays the highest, lowest and total salary paid to the employees in that specific department in tabular format. Consider following tables for this.

```
dept(deptno, deptname, location)
emp (empno, empname, addr, phno, sal, deptno)
```

Solution:

disp_emp.html

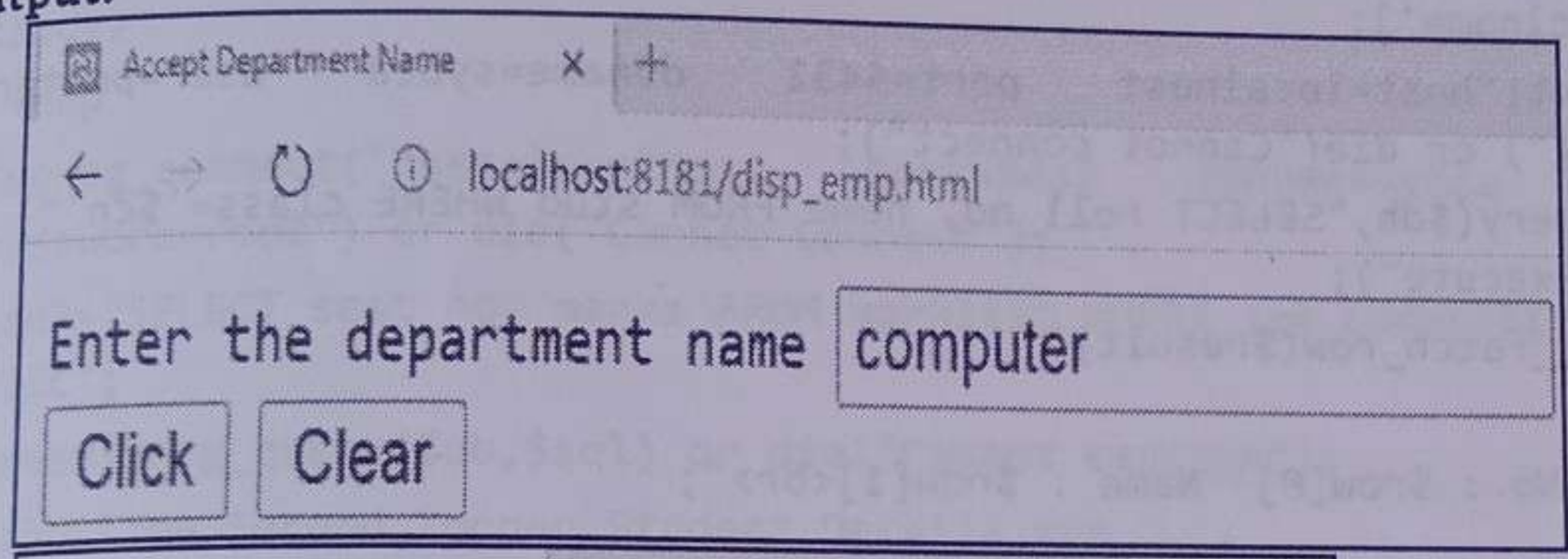
```
<html>
<head><title>Accept Department Name</title></head>
<body>
<form action="disp_emp.php" method="POST">
<pre>
Enter the department name <input type="text" name="dn">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

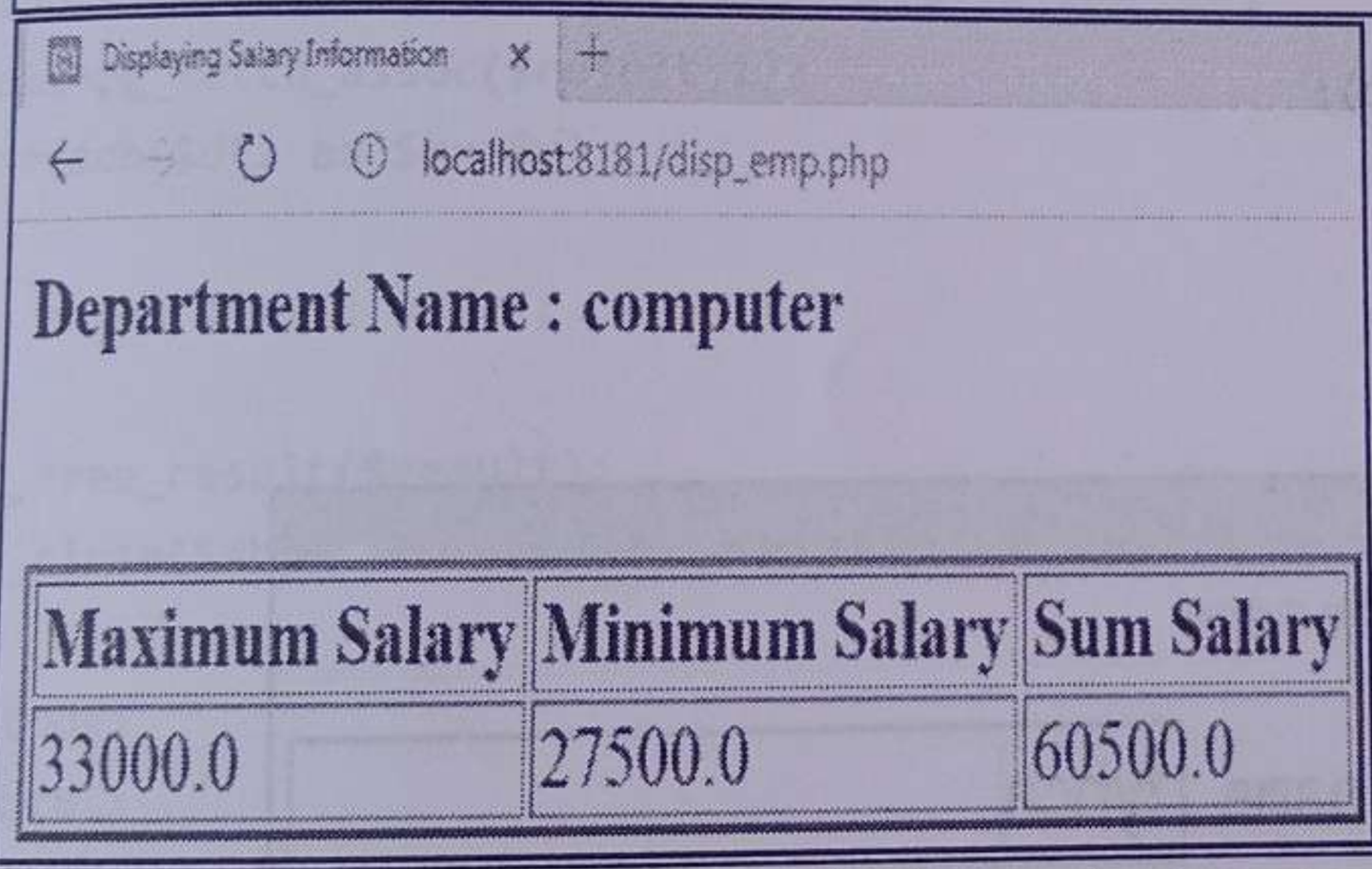
disp_emp.php

```
<html>
<head><title>Displaying Salary Information</title></head>
<body>
<?php
$inp=$_GET['dn'];
$db=pg_connect("host=localhost   port=5432   dbname=sybca   user=postgres
password=root") or die("Unable to connect");
$sql="SELECT  MAX(EMP.SAL),MIN(EMP.SAL),SUM(EMP.SAL)  FROM  DEPT,EMP  WHERE
DEPT.DEPTNO=EMP.DEPTNO AND DEPT.DEPTNAME='$inp';";
echo "<h4>Department Name : $inp</h4><br>";
echo "<table border='2'>";
echo "<tr><th>Maximum Salary</th><th>Minimum Salary</th>
      <th>Sum Salary</th></tr>";
$result=pg_query($db,$sql) or die("Cannot Execute");
while($data=pg_fetch_row($result))
{
echo "<tr><td>$data[0]</td><td>$data[1]</td><td>$data[2]</td></tr>";
}
```

```
echo "</table>";
pg_free_result($result);
pg_close($db);
?>
</body>
</html>
```

Output:





Program 5.23: Write a PHP program which accepts class name from user and displays the roll_no and name of the students studying in that class. Consider stud (roll_no, name, class) table.

Solution:

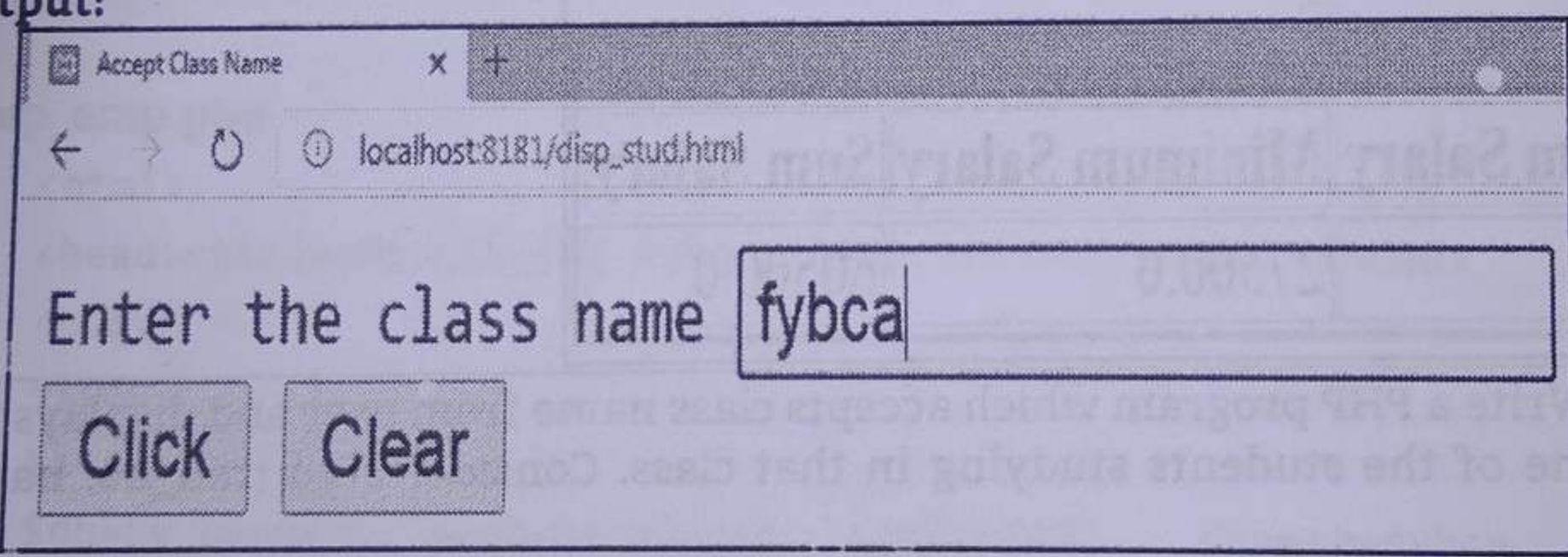
disp_stud.html

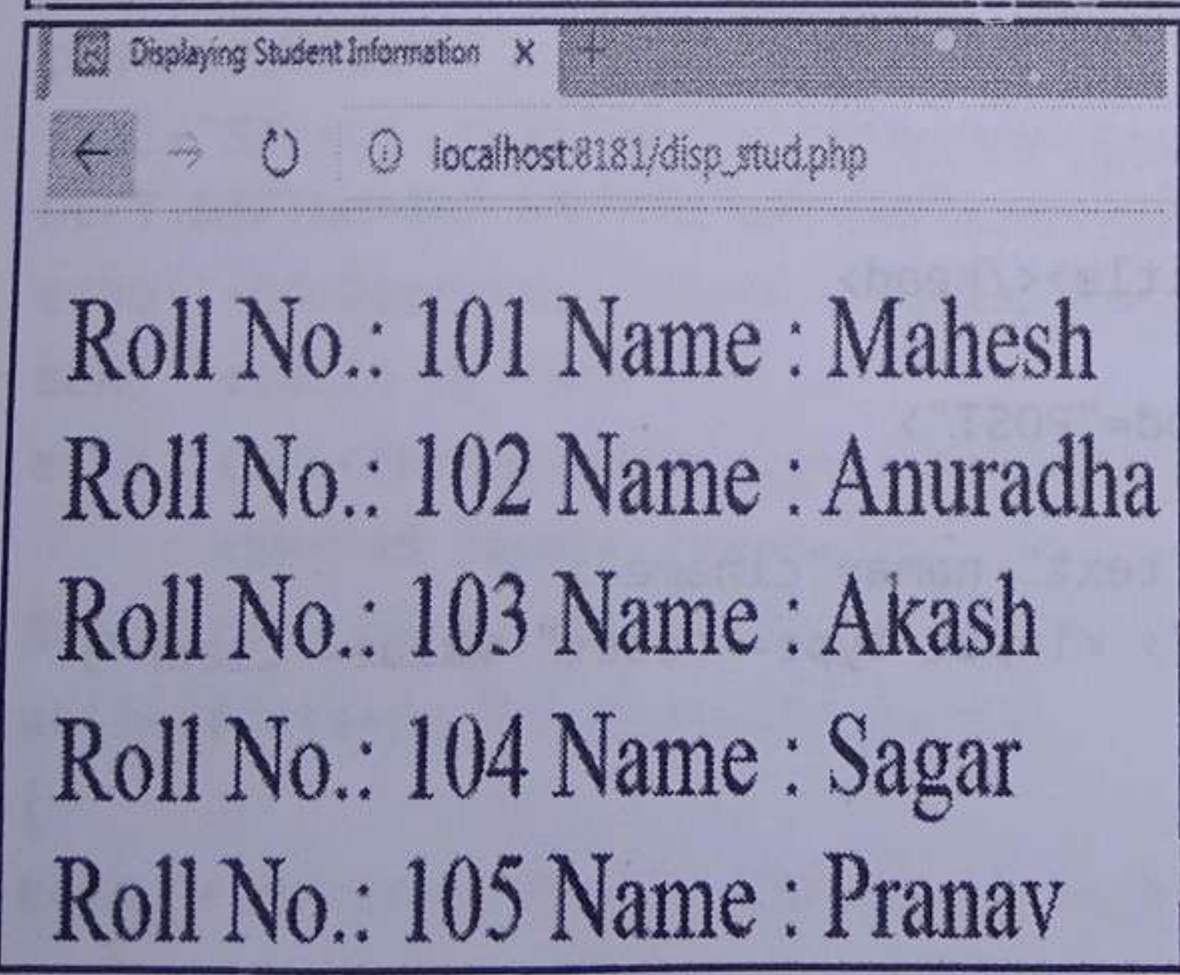
```
<html>
<head><title>Accept Class Name</title></head>
<body>
<form action="disp_stud.php" method="POST">
<pre>
Enter the class name <input type="text" name="clname">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

disp_stud.php

```
<html>
<head><title>Displaying Student Information</title></head>
<body>
<?php
$cn=$_POST['clname'];
$db=pg_connect("host=localhost    port=5432    dbname=sybca    user=postgres
password=root") or die("Cannot connect");
$result=pg_query($db,"SELECT roll_no, name FROM stud WHERE class='$cn'") or
die("Cannot Execute");
while($row=pg_fetch_row($result))
{
  echo "Roll No.: $row[0]  Name : $row[1]<br>";
}
pg_free_result($result);
pg_close($db);
?>
</body>
</html>
```

Output:





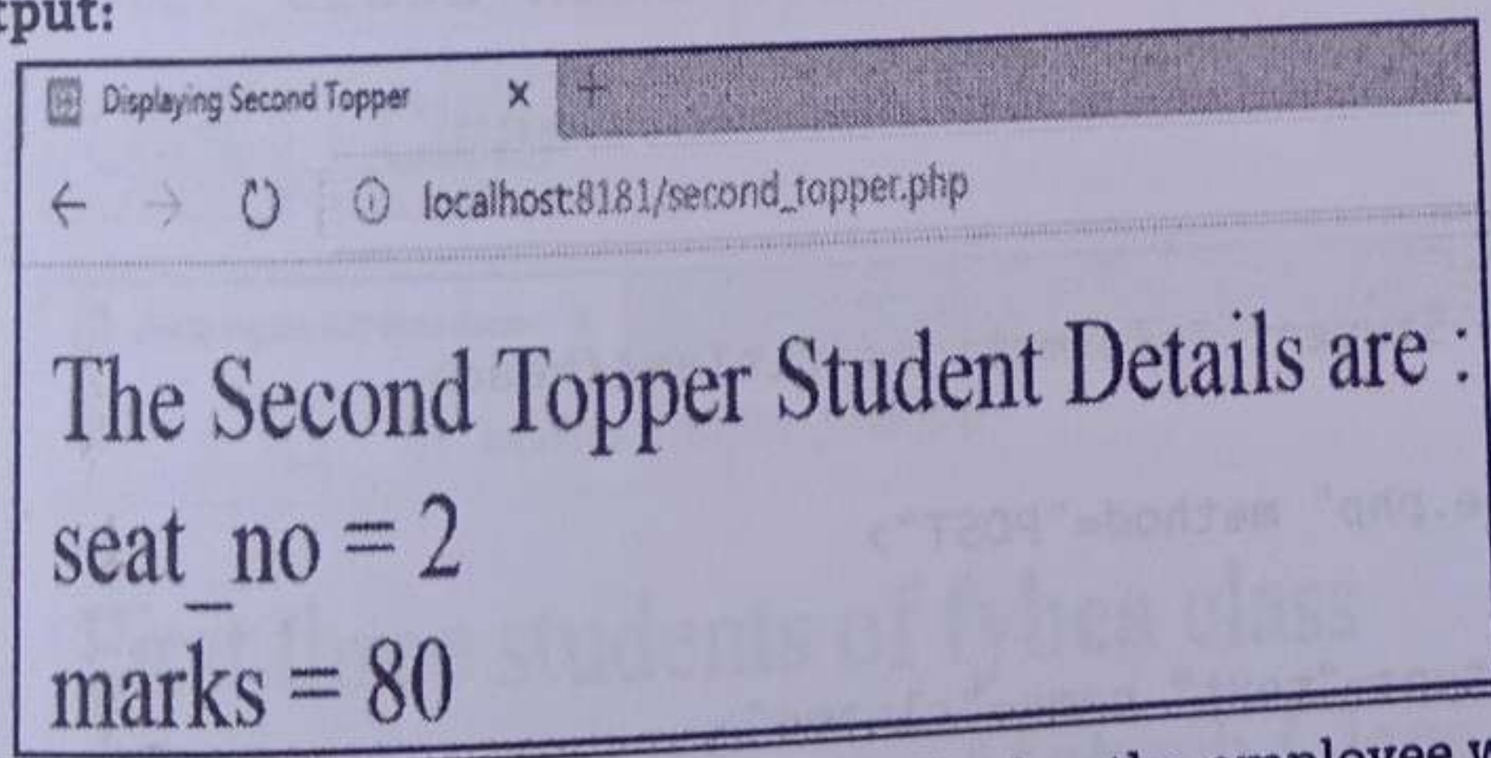
Program 5.24: Consider a table marklist(seat_no, sub_code, marks). Write a PHP program that will display the marks of the second topper student in sub_code=101.

Solution:

second_topper.php

```
<html><head><title>Displaying Second Topper</title></head>
<body>
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
password=root") or die("Cannot Connect");
$sql="SELECT seat_no, marks FROM marklist WHERE sub_code=101 ORDER BY marks
DESC";
$result=pg_query($db,$sql) or die("Cannot execute");
echo "The Second Topper Student Details are : ";
$dtl=pg_fetch_assoc($result,1);
foreach($dtl as $k=>$v)
{
   echo "<br>$k = $v";
}
pg_free_result($result);
pg_close($db);
?>
</body>
</html>
```

Output:



Program 5.25: Write a PHP script to display the employee whose salary is greater than or equal to average salary.

Consider following tables for this.

```
dept(deptno, deptname, location)
emp (empno, empname, addr, phno, sal, deptno)
```

Solution:

emp_sal.php

```
<html><head><title>Employee having sal>=avg(sal)</title></head>
<body>
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
    password=root") or die("Cannot Connect");
$sql="SELECT empname FROM emp WHERE sal>=(SELECT avg(sal) FROM emp);";
$result=pg_query($db,$sql) or die("Cannot Execute");
while($row=pg_fetch_row($result))
{
   echo "<br>Name : ".$row[0];
}
pg_free_result($result);
pg_close($db);
?>
</body>
</html>
```

Output:

Name: Rajesh

Name: Abhijeet

Program 5.26: Write the PHP script to display the details of first 3 students of a user specified class according to their roll numbers. Consider the table stud(roll_no, name, class).

Solution:

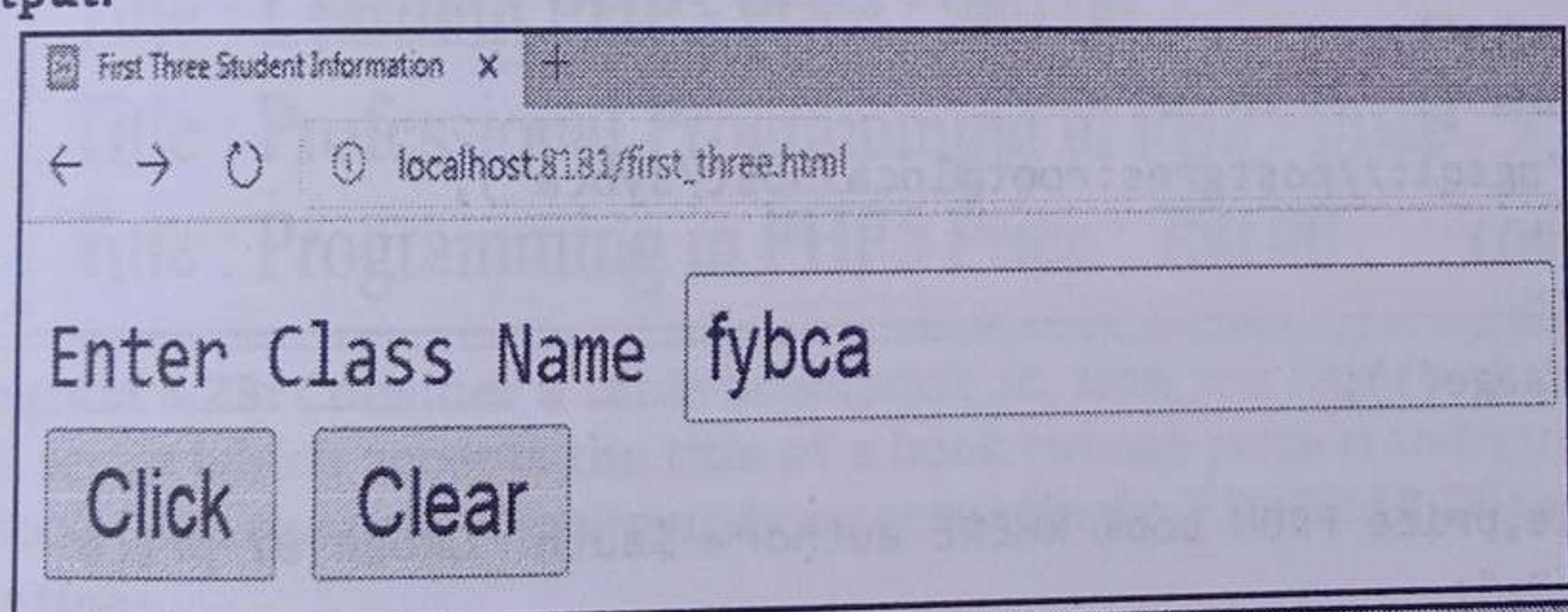
first_three.html

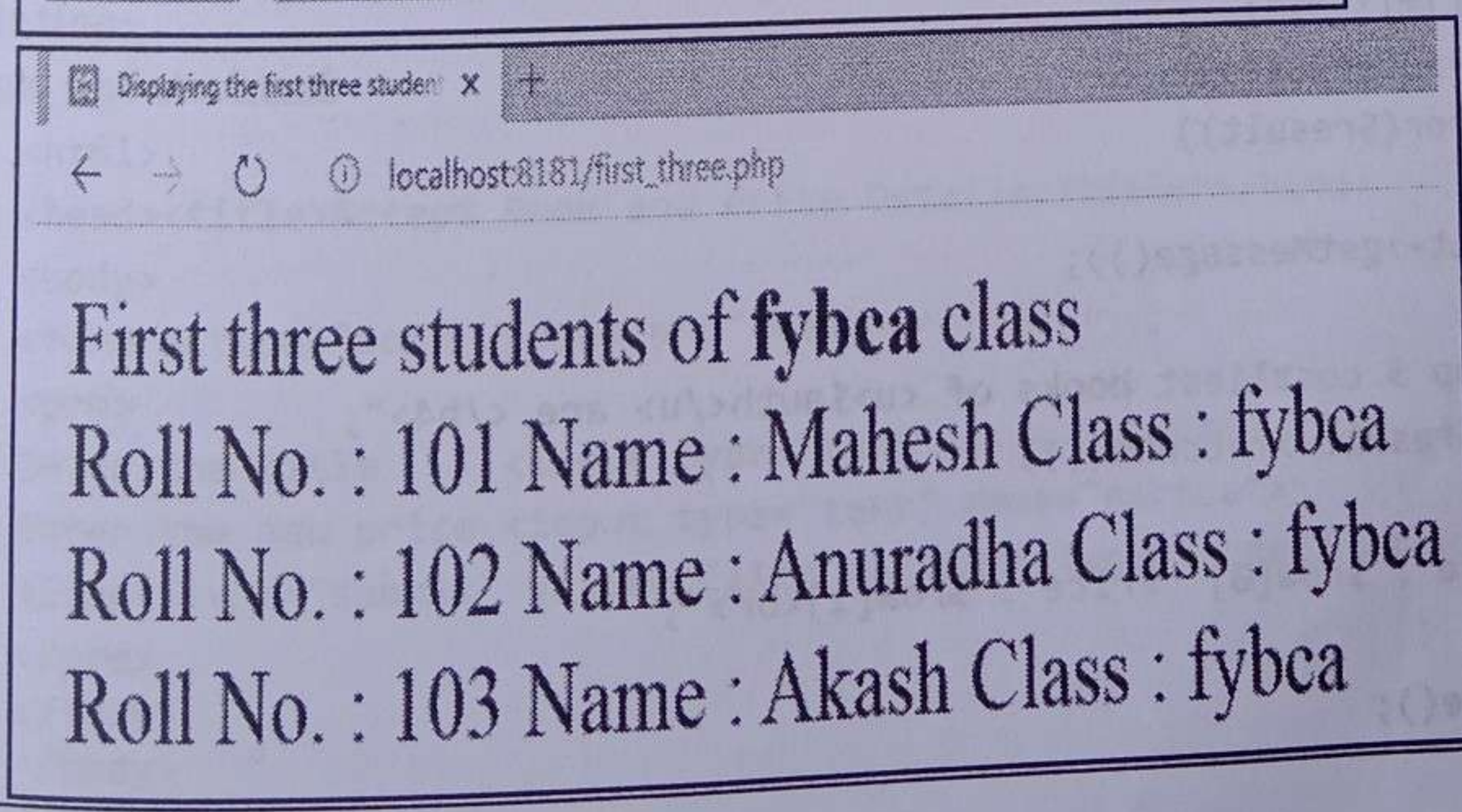
```
<html>
<head><title>First Three Student Information</title></head>
<body>
<form action="first_three.php" method="POST">
<pre>
Enter Class Name <input type="text" name="clname">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

first_three.php

```
<html><head><title>Displaying the first three students</title></head>
<body>
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
    password=root") or die("Cannot Connect");
$inp=$_POST['clname'];
$sql="SELECT * FROM stud WHERE class='$inp' LIMIT 3";
$result=pg_query($db,$sql) or die("Cannot Execute");
echo "First three students of <b>$inp</b> class";
while($row=pg_fetch_row($result))
{
  echo "<br>Roll No. : $row[0]  Name : $row[1]  Class : $row[2]";
}
pg_free_result($result);
pg_close($db);
?>
<body>
</html>
```

Output:





Program 5.27: Consider a table book(book_id, isbn_no, title, author, publ, price). Write a PHP script to display the top 3 costliest book written by user specified author name (Use PEAR DB method).

Solution:

top_three.html

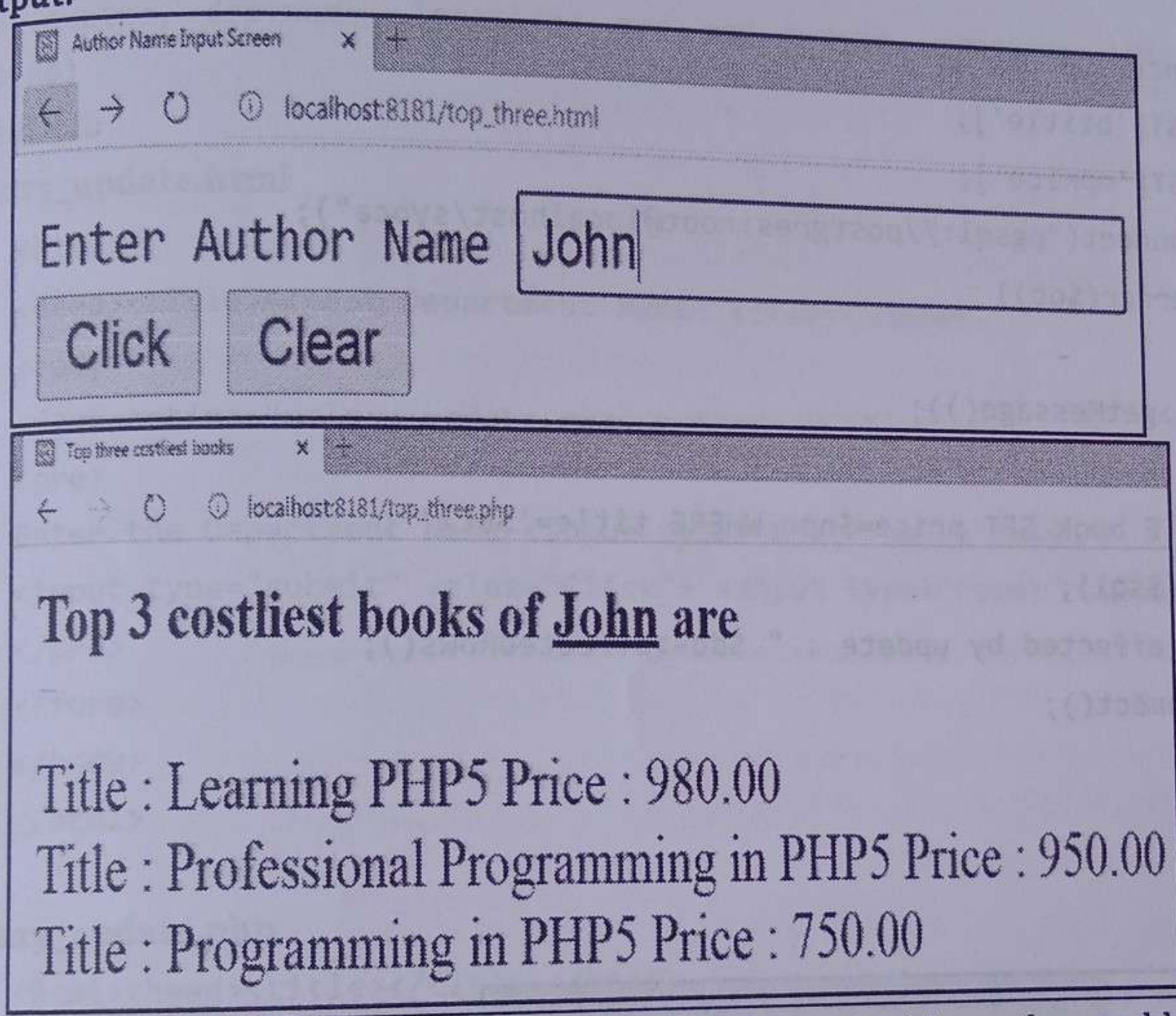
```
<html>
<head><title>Author Name Input Screen</title></head>
<body>
<form action="top_three.php" method="POST">
<pre>
Enter Author Name <input type="text" name="aname">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

top_three.php

```
<html><head><title>Top three costliest books</title></head>
<body>
<?php
require_once('DB.php');
$auth=$_POST['aname'];
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
  die($db->getMessage());
}
$sql="SELECT title,price FROM book WHERE author='$auth' ORDER BY price
      DESC LIMIT 3;";
$result=$db->query($sql);
if(DB::iserror($result))
{
  die($result->getMessage());
}
echo "<h4>Top 3 costliest books of <u>$auth</u> are </h4>";
while($row=$result->fetchRow())
{
  echo "Title : $row[0]  Price : $row[1]<br>";
}
$result->free();
```

```
$db->disconnect();
?>
</body>
</html>
```

Output:



Program 5.28: Consider a table book(book_id, isbn_no, title, author, publ, price). Write a PHP script which accepts the title of a book (whose price is to be updated) and new price of book from the user. Do the updates accordingly. (Use PEAR DB method).

Solution:

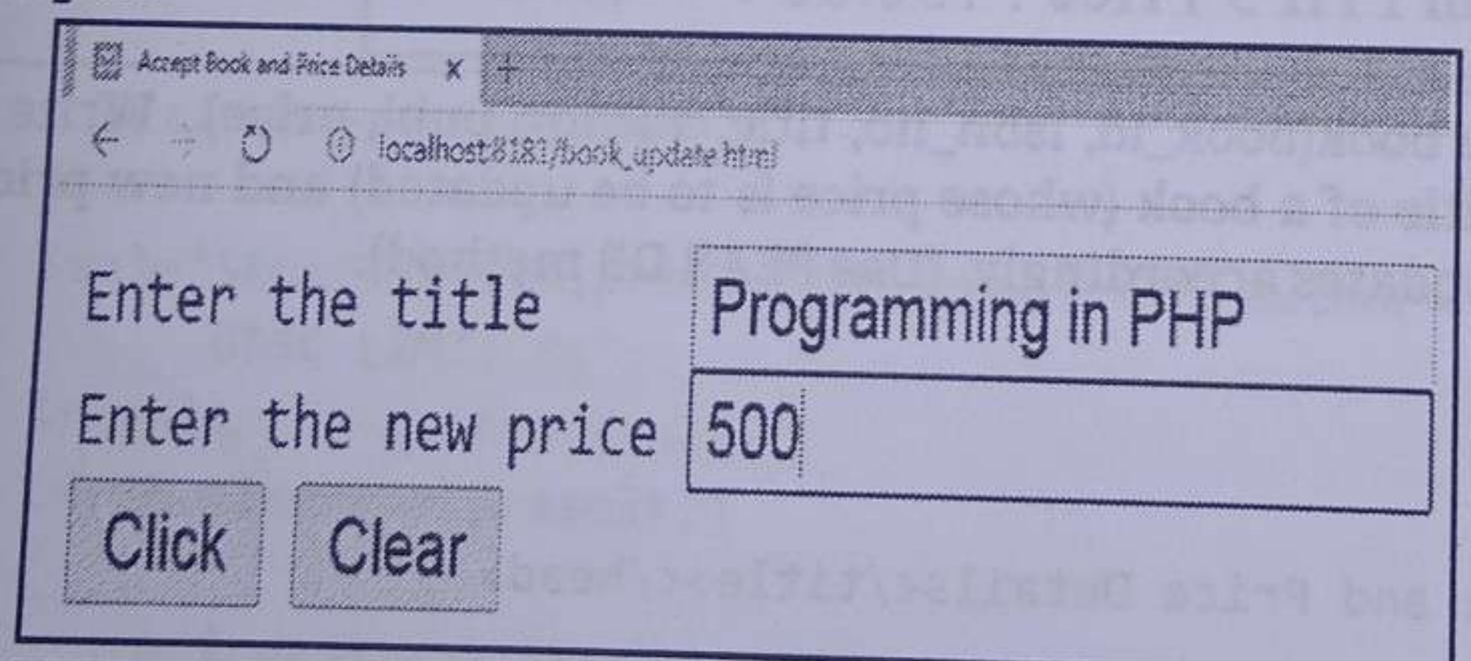
book_update.html

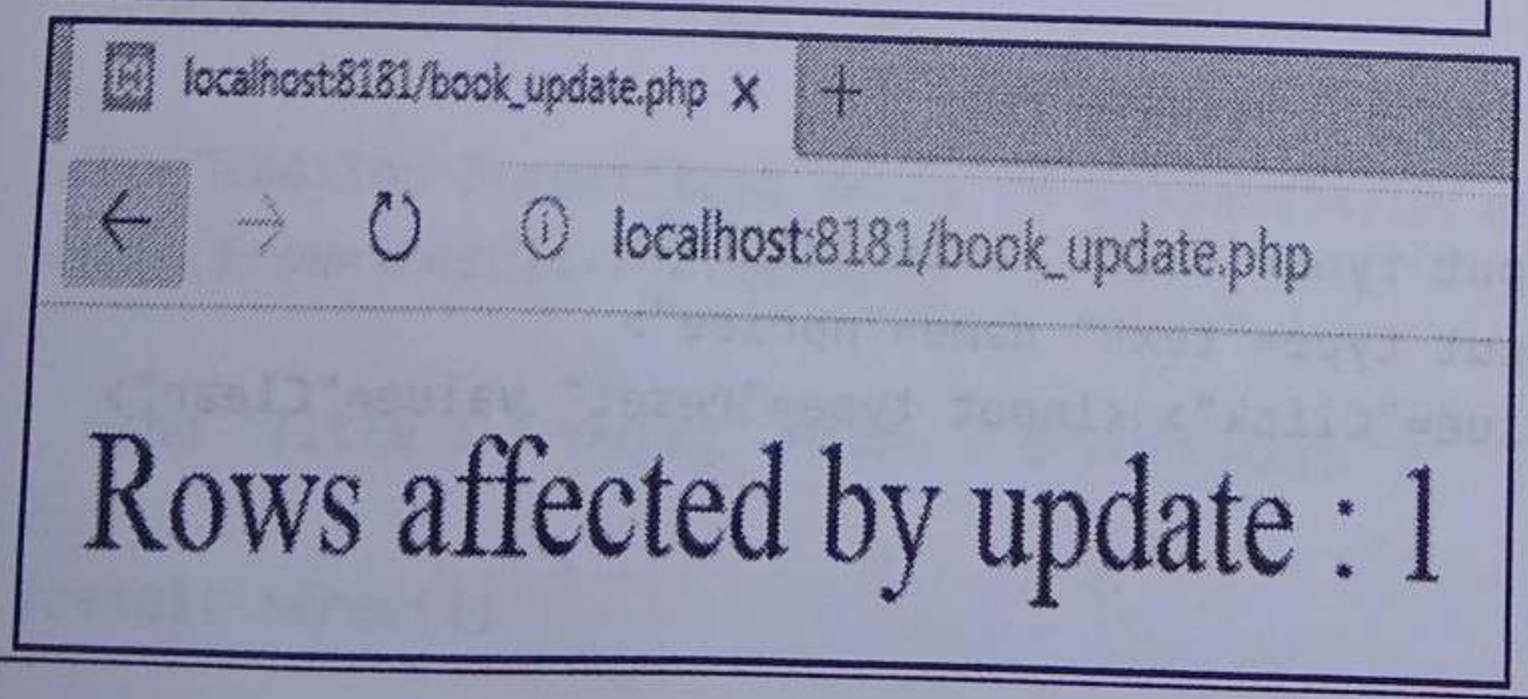
```
<html>
<head><title>Accept Book and Price Details</title></head>
<body>
<form action="book_update.php" method="POST">
<pre>
Enter the title      <input type="text" name="btitle">
Enter the new price <input type="text" name="nprice">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

book_update.php

```
<html><head><title></title></head>
<body>
<?php
require_once('DB.php');
$btl=$_POST['btitle'];
$npr=$_POST['nprice'];
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
  die($db->getMessage());
}
$sql="UPDATE book SET price=$npr WHERE title='$btl'";
$db->query($sql);
echo "Rows affected by update : ".$db->affectedRows();
$db->disconnect();
?>
</body>
</html>
```

Output:





Program 5.29: Write a PHP script to accept the department name from the user. Increase the salaries of the employees working in that department by 10% and also show the number of employees benefited by this. Consider the following tables and use PEAR DB method.

dept(deptno, deptname, location) and emp (empno, empname, addr, phno, sal, deptno)

Solution:

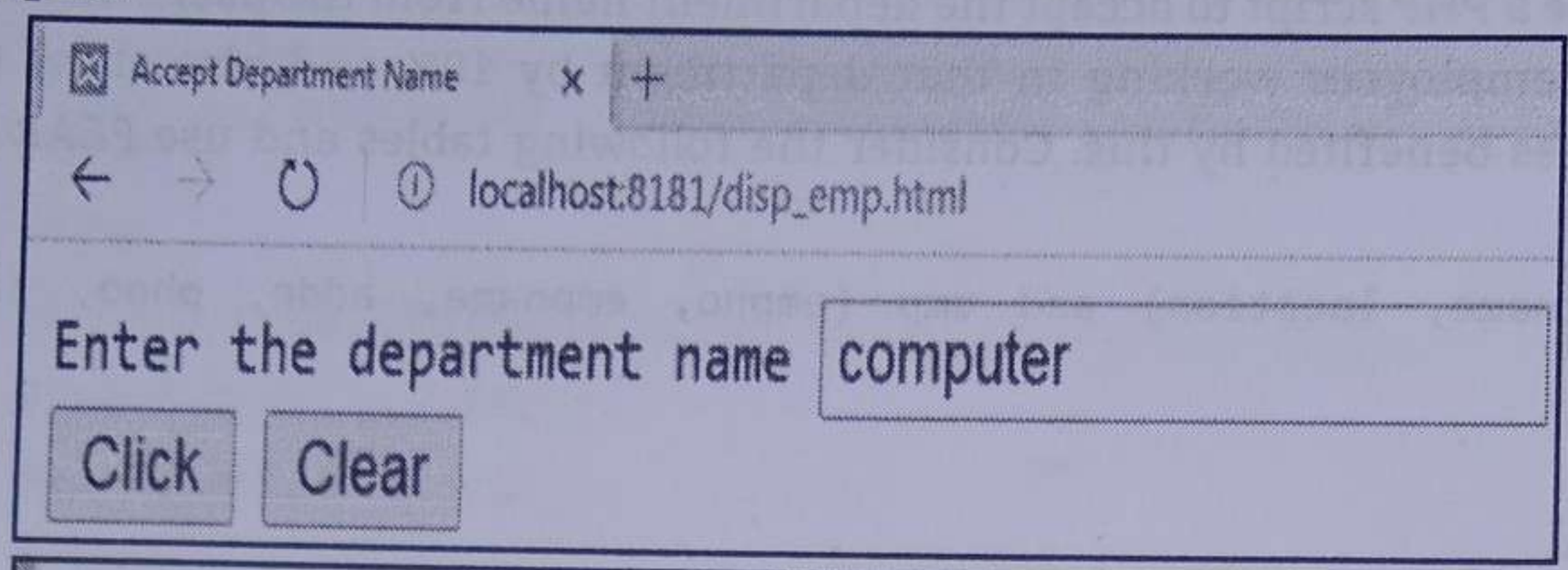
salary_update.html

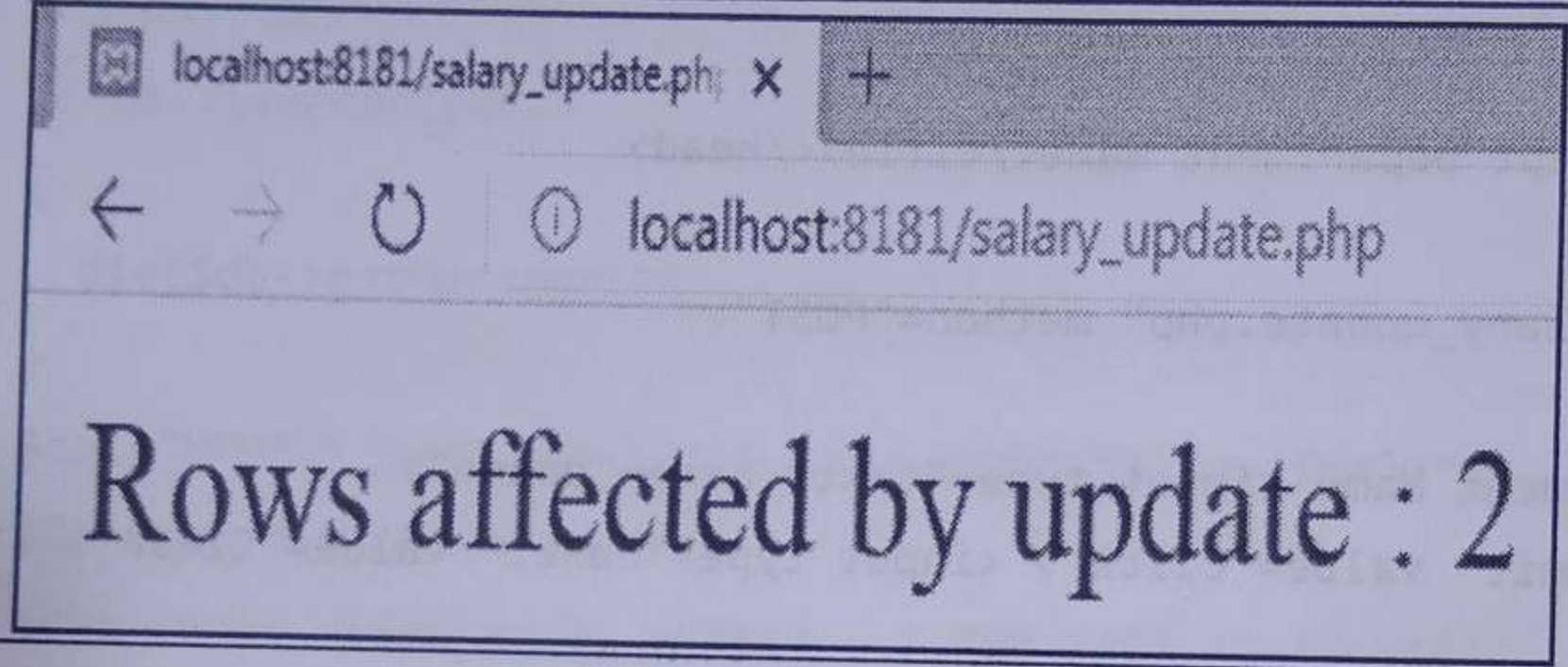
```
<html>
<head><title>Accept Department Name</title></head>
<body>
<form action="salary_update.php" method="POST">
<pre>
Enter the Department Name <input type="text" name="dname">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

salary_update.php

```
<html><head><title></title></head>
<?php
require_once('DB.php');
$dn=$_POST['dname'];
$db=DB::connect("pgsql://postgres:root@localhost/sybca");
if(DB::iserror($db))
{
  die($db->getMessage());
}
$sql="UPDATE emp SET sal=sal*1.1 WHERE deptno IN (SELECT deptno FROM dept
WHERE deptname='$dn')";
$db->query($sql);
echo "Rows affected by update : ".$db->affectedRows();
$db->disconnect();
?>
</body></html>
```

Output:





Program 5.30: Write a PHP script which accepts the number of employees as an input from user and display the department names which has total number of employees greater than or equal to the value entered by the user (Use database specific method). Consider following tables for this.

```
dept(deptno, deptname, location) emp(empno, empname, addr, phno, sal, deptno)
```

Solution:

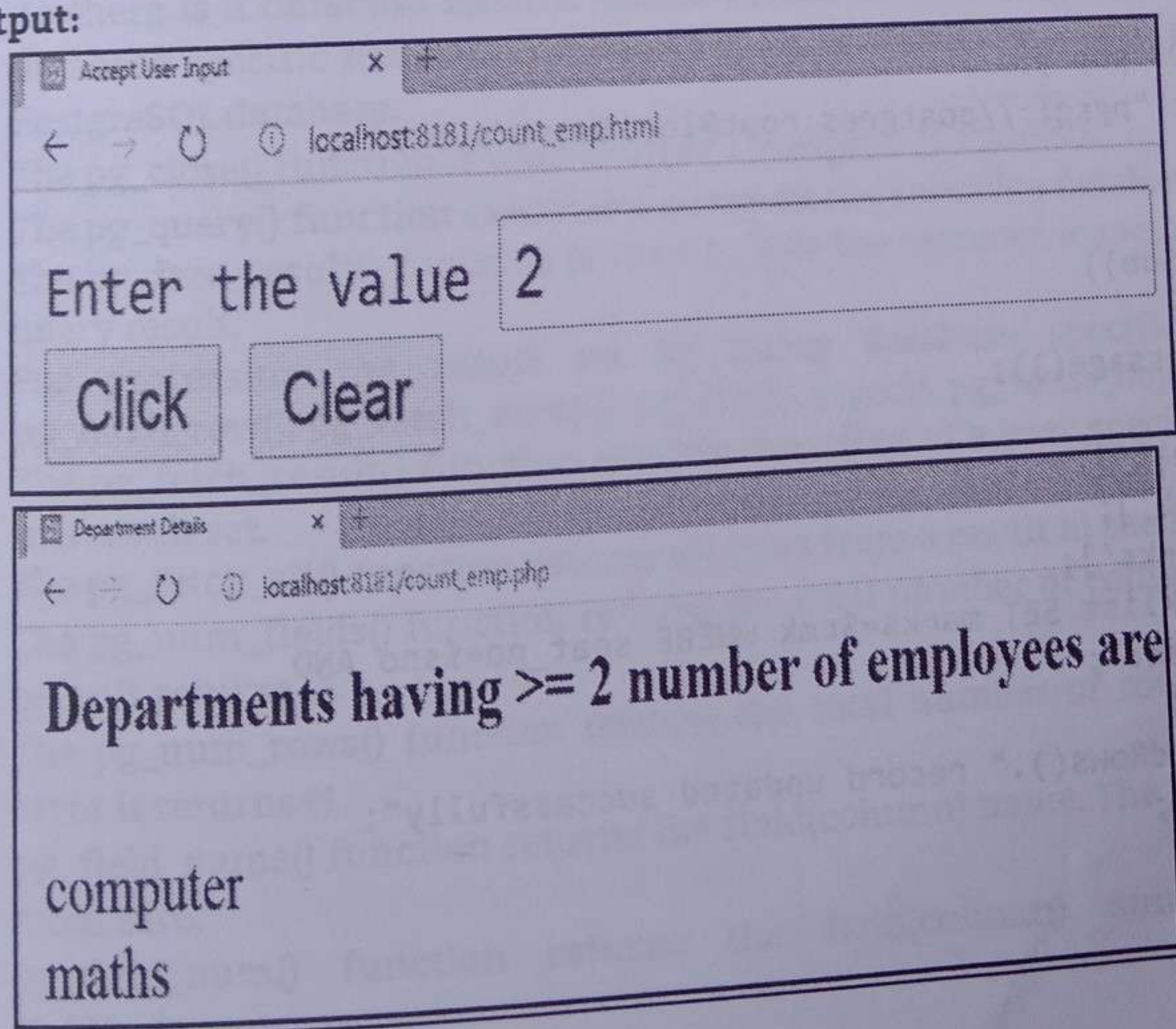
count_emp.html

```
<html>
<head><title>Accept User Input</title></head>
<body>
<form action="count_emp.php" method="POST">
<pre>
Enter the value <input type="text" name="tnemp">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

count_emp.php

```
<html>
<head><title>Department Details</title></head>
<body>
<?php
$db=pg_connect("host=localhost port=5432 dbname=sybca user=postgres
    password=root") or die("Cannot Connect");
$inp=$_POST['tnemp'];
$sql="SELECT deptname FROM dept WHERE deptno IN(SELECT deptno FROM emp
    GROUP BY deptno HAVING COUNT(*)>=$inp)";
$result=pg_query($db,$sql) or die("Cannot Execute");
echo "<h4>Departments having >= $inp number of employees are </h4>";
while($data=pg_fetch_row($result))
{
  echo $data[0]."<br>";
}
pg_free_result($result);
pg_close($db);
?>
</body>
</html>
```

Output:



Program 5.31: Consider a table marklist(seat_no, sub_code, marks). Write a PHP script which to accept seat_no, sub_code and updated marks from user and modify the marks of the student to changed marks. (Use PEAR DB method).

Solution:

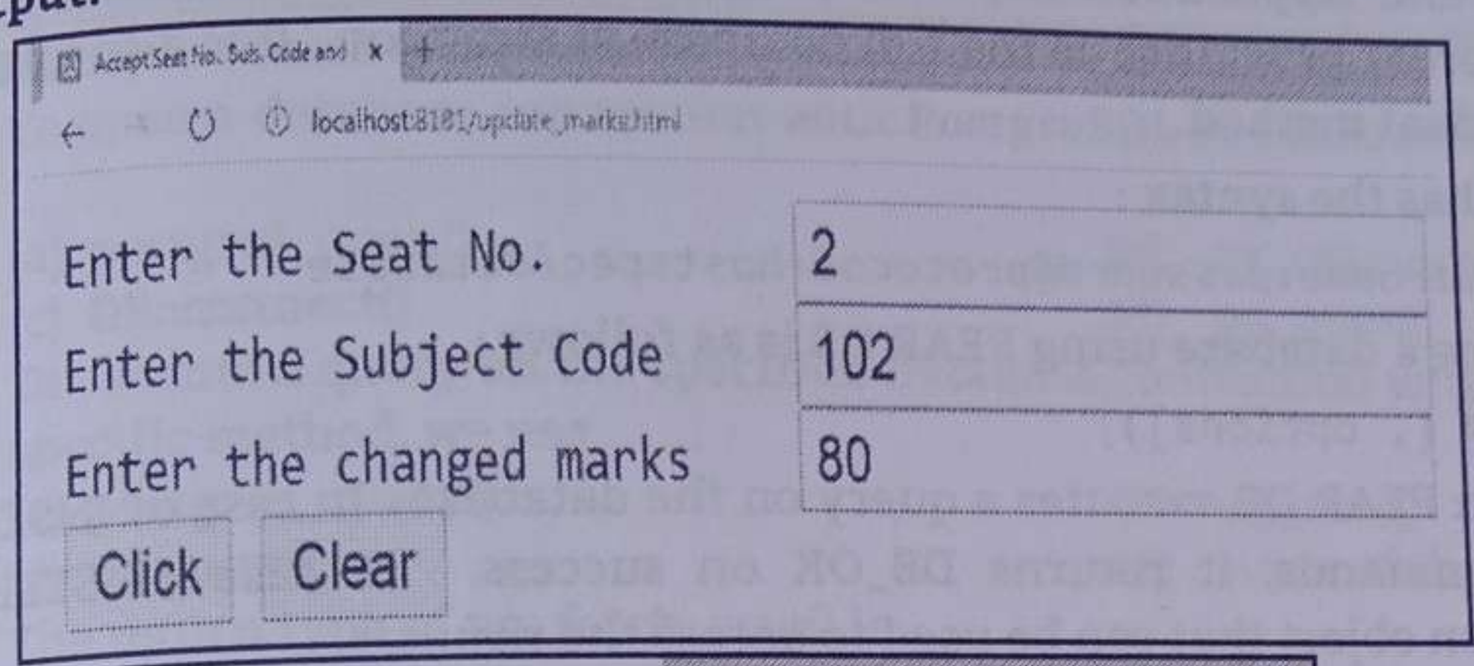
update_marks.html

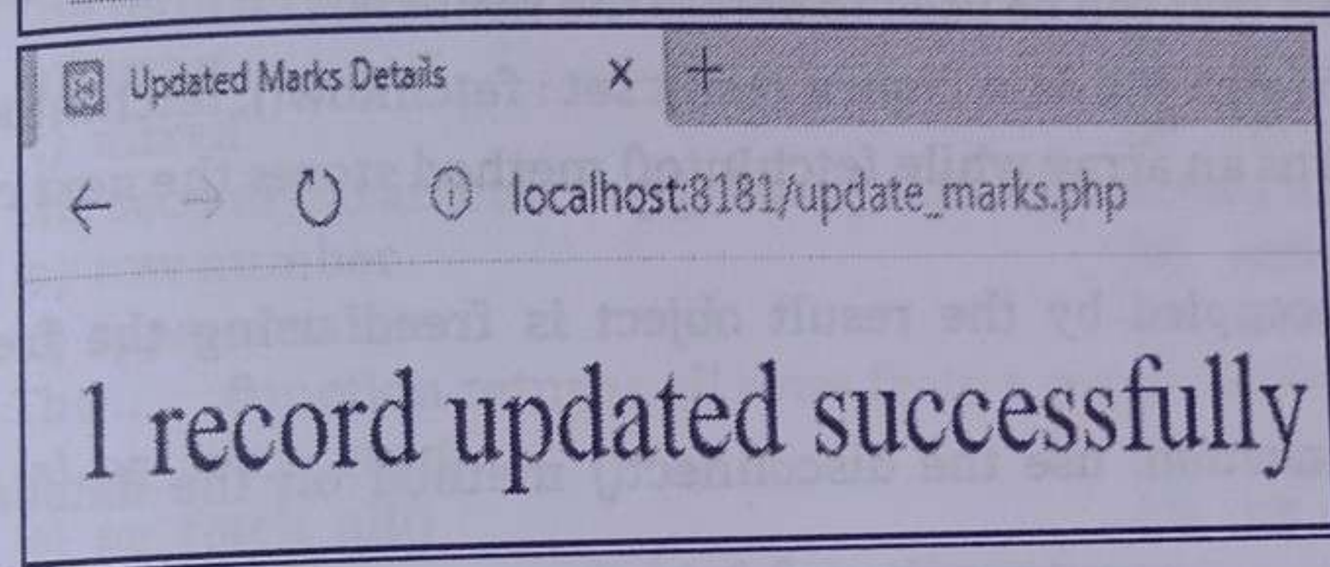
```
<html>
<head><title>Accept Seat No., Sub. Code and Marks</title></head>
<body>
<form action="update_marks.php" method="POST">
<pre>
Enter the Seat No.          <input type="text" name="tseat">
Enter the Subject Code      <input type="text" name="tsub">
Enter the changed marks     <input type="text" name="tmarks">
<input type="submit" value="Click"> <input type="reset" value="Clear">
</pre>
</form>
</body>
</html>
```

update_marks.php

```
<html>
<head><title>Updated Marks Details</title></head>
<body>
<?php
require_once('DB.php');
$db=DB::connect("pgsql://postgres:root@localhost/sybca");


if(DB::iserror($db))
{
  die($db->getMessage());
}
$sno=$_POST['tseat'];
$sco=$_POST['tsub'];
$cmk=$_POST['tmarks'];
$sql="UPDATE marklist SET marks=$cmk WHERE seat_no=$sno AND
      sub_code=$sco";
$db->query($sql);
echo $db->affectedRows()." record updated successfully";
$db->disconnect();
?>
</body>
</html>
```

Output:





Summary

- There are two methods to access databases from PHP : database specific method, database independent method (PEAR DB).
- As there is a database specific prefix before the function, it is easy to identify the database specific method. For example, pg_connect() function is used to connect to PostgreSQL database.
- The pg_close() function is used to close a PostgreSQL connection.
- The pg_query() function executes a query on the specified database connection.
- The pg_free_result() function is used to free the memory associated with PostgreSQL query result.
- For processing the result set by using database specific method we have pg_fetch_row(), pg_fetch_array(), pg_fetch_assoc(), pg_fetch_object() functions.
- The pg_fetch_result() function returns the value of a user specified row and column in a result set.
- The pg_fetch_all() function returns all rows from a result in the form of an array.
- The pg_num_fields() function returns the total number of columns in a result set. On error it returns -1.
- The pg_num_rows() function returns the total number of rows in a result set. On error it returns -1.
- pg_field_name() function returns the field(column) name. The field numbering starts from zero.
- pg_field_num() function returns the field(column) number of the named field(column).

- The PHP Extension and Application Repository (PEAR) DB is an advanced, object-oriented database library providing us the full database abstraction. It is also known as database independent method.
- The DSN in PEAR DB has the syntax :

```
type(dbsyntax)://username:password@protocol+hostspec/database
```

- The syntax for opening a database using PEAR DB is as follows :

```
$db=DB::connect(DSN [, options]);
```

- The query() method in PEAR DB executes a query on the database. In case of INSERT, UPDATE, DELETE commands, it returns DB_OK on success. In case of SELECT command, it returns an object that can be used to access the result.
- PEAR DB has two methods to fetch the data from a result set : fetchRow(), fetchInto().
- The fetchRow() method returns an array while fetchInto() method stores the next row in an array.
- In PEAR DB, the memory occupied by the result object is freed using the free() method.
- To release the database connection, use the disconnect() method on the database object.
- If we want to execute the same query repeatedly, it is always better to compile the query first and then execute the compiled query many times. This can be achieved by using the prepare(), execute() and executeMultiple() methods.
- The getOne() method executes a query and returns the first column of the first row.
- The getCol() method returns a single column from the result returned by the SQL query.
- The getRow() method returns the first row of result data returned by the SQL query.
- The getAssoc() method returns the whole result set in the form of an associative array.
- The getAll() method returns an array containing all the rows returned by the SQL query. If we specify the placeholder in the SQL statement, it retrieves only specific rows.
- The numRows() method returns the total number of rows returned by the SELECT query.
- The numCols() method returns the total number of columns returned by the SELECT query.
- The affectedRows() method gives us information about the number of rows affected by UPDATE, INSERT or DELETE statements.
- The getListOf() method gives information on available databases, users, views and functions.
- PEAR DB offers commit() and rollback() methods to help with transactions.
- A sequence is a table in the database that maintains the last-assigned ID. We can create and delete sequences with createSequence() and dropSequence() methods.

Check Your Understanding

Q.I Multiple choice questions.

1. To open a database connection with PostgreSQL in database specific method, we use .....
   (a) postgres_open() (b) pg_connect()
   (c) DB::connect() (d) pg_open()
2. To execute a query on the specified database connection with PostgreSQL in database specific method, we use .....
   (a) postgres_query() (b) pg_run()
   (c) pg_query (d) none of these
3. The return type of pg_fetch_row() is .....
   (a) array (b) string
   (c) mixed (d) integer
4. The second parameter to pg_fetch_object() function is the ..... in the result to fetch.
   (a) row number (b) column number
   (c) type of value (d) none of these
5. The ..... function returns all rows from a result in the form of an array.
   (a) pg_fetch_row() (b) pg_fetch_array()
   (c) pg_fetch_all() (d) pg_fetch_object()
6. On error, the pg_num_fields() function returns .....
   (a) 0 (b) 1
   (c) –1 (d) none of these
7. The ..... function returns the column name.
   (a) pg_field_name() (b) pg_col_name()
   (c) postgres_field_name() (d) none of these
8. The ..... function free the memory occupied by the PostgreSQL query result.
   (a) pg_free_result() (b) pg_free()
   (c) pg_result_free() (d) none of these
9. The ..... function is used to close a PostgreSQL conenction.
   (a) pg_disconnect() (b) pg_free()
   (c) pg_conn_close() (d) pg_close()
10. The ..... function returns the value of a user specified row and column in a result set.
    (a) pg_fetch_row() (b) pg_fetch_result()
    (c) pg_fetch_assoc() (d) pg_fetch_array()
11. The ..... function returns the total number of rows in a result set.
    (a) pg_tot_rows() (b) pg_num_rows()
    (c) pg_num_records() (d) pg_total_rows()
12. The ..... function displays the error message related with the result.
    (a) pg_result_error() (b) pg_error()
    (c) pg_error_show() (d) none of these
13. To execute a query on the database using PEAR DB, we use ..... method.
    (a) query() (b) pg_query()
    (c) execute() (d) pg_execute()

14. When we execute INSERT, UPDATE, DELETE queries, the query() method returns the ..... constant to indicate success.
    (a) DB_SUCCESS (b) DB_OK
    (c) DB_DONE (d) none of these
15. The ..... method in PEAR DB returns an array of the next row of results.
    (a) fetchRow() (b) pg_fetch_row()
    (c) pg_fetch_array() (d) fetchArray()
16. The fetchRow() method returns ..... if there is no more data.
    (a) DB_ERROR (b) NULL
    (c) DB_FAIL (d) none of these
17. The default mode which controls the format of the row array in fetchRow() method is .....
    (a) DB_FETCHMODE_ASSOC (b) DB_FETCHMODE_ORDERED
    (c) DB_FETCHMODE_OBJECT (d) none of these
18. When the processing on the result object is over, the memory occupied by it is released using the ..... method in PEAR DB.
    (a) free() (b) close()
    (c) disconnect() (d) pg_free_result()
19. The ..... method executes a query and returns the first column of the first row.
    (a) getCol() (b) getAssoc()
    (c) getOne() (d) getRow()
20. The ..... method returns the total number of rows returned by the SELECT query in PEAR DB.
    (a) numRecords() (b) numRows()
    (c) pg_num_rows() (d) num_Rows()
21. The ..... method query the database and gives information on available databases, users, views, and functions.
    (a) getInfo() (b) getListOf()
    (c) tableInfo() (d) none of these

ANSWERS

1. (b)	2. (c)	3. (a)	4. (a)	5. (c)	6. (c)	7. (a)	8. (a)
9. (d)	10. (b)	11. (b)	12. (a)	13. (a)	14. (b)	16. (b)	17. (b)
18. (a)	19. (c)	20. (b)	21. (b)				

Q.II State True/False.

1. In a database specific method, there are specific database prefixes used before the connect() function.
2. If the first parameter to pg_query() function is omitted, the default connection is used.
3. The second parameter to pg_fetch_row() function is the row number in the result to fetch.
4. The pg_fetch_array() function returns the row as an array only in indexed format.

5. The pg_fetch_assoc() function is equivalent to pg_fetch_array() with PGSQL_ASSOC as the optional third parameter.
6. If called with single parameter, each call to pg_fetch_object() results in error.
7. The pg_num_fields() function returns the total number of columns in a result set.
8. By using DB::iserror() method, we can check whether there is an error or not.
9. The fetchRow() method returns an array while fetchInto() method stores it into an array.
10. The DB_FETCHMODE_ORDERED creates an associative array.
11. The disconnect() method is not mandatory, as all the database connections are terminated when the PHP script ends.
12. While writing the query like INSERT or UPDATE, we can mention a placeholder '%' in place of specific value.
13. If we want to execute the same query repeatedly, then it is always better to compile the query once and then execute the compiled query many times.
14. The getCol() method returns all columns from the result returned by the SQL query.
15. The getRow() method returns the first row of result data returned by the SQL query.
16. The numCols() method returns the array of all the columns returned by the SQL query.
17. The affectedRows() method gives us the information about the number of rows affected by UPDATE, INSERT or DELETE statements.
18. The tableInfo() method returns the detail information on the data type and other points like flags of columns returned from a SELECT operation.
19. A sequence is a table in the database that maintains the last-assigned ID.

ANSWERS

1. True	2. True	3. True	4. False	5. True	6. False	7. True	8. True
9. True	10. False	11. True	12. False	13. True	14. False	15. True	16. False
17. True	18. True	19. True					

Practice Questions

Q.I Answer the following questions in short.

1. What is a backend? List the different databases supported by PHP.
2. State the syntax and purpose of pg_connect() function.
3. State the syntax and purpose pg_query() function.
4. State the difference between pg_fetch_row() and pg_fetch_array() function.
5. State the example in which we can use pg_fetch_result() function.
6. What is the difference between pg_fetch_all() and pg_fetch_assoc() functions.
7. For what purpose pg_result_error() function is used?
8. What is the difference between database specific method and PEAR DB?
9. Compare between pg_connect() and DB::connect() method.
10. Compare between pg_query() and query() method.

11. For what purpose fetchInto() method is used?
12. List the different possible values of mode parameter to fetchInto() that controls the format of the row array.
13. Write in short about free() method.
14. Write in short about disconnect() method.
15. List the characters that can be used as placeholder values in an SQL query.
16. Write in short about prepare() method.
17. What is the difference between getOne() and getCol() method?
18. What is the difference between numRows() and numCols() methods?
19. For what purpose tableInfo() method is used?

Q.II Answer the following questions.

1. What are the steps to open and interact with a database in PHP?
2. Explain the processing of result set in database specific method.
3. Explain with example about pg_fetch_result() function.
4. Explain with example about pg_fetch_all() function.
5. Explain with example about pg_field_name() and pg_field_num() functions.
6. How to fetch the results from a query using PEAR DB? Explain with suitable example.
7. What are the placeholders in a query?
8. What is the advantage of using Prepare/Execute method?
9. Consider tables: game(gno, gname) , player(pno, pname, age), g_p(gno, pno, score, rank)
   Write a PHP script to accept a score from user and display all the players who are playing "cricket" having a score greater than or equal to the value entered by user.
10. Consider table : plant(plant_id, plant_name, plant_type, price)
    Write a PHP script to accept the plant_type and price from user. The script should display the information of plants belonging to the plant_type entered by user and having price greater than or equal to the price entered by the user.
11. Consider the plant table given in Question 10. Write a PHP script to accept plant_name and updated price of that plant from user. Using PHP script updates the price of the plant entered by the user.
12. How to get information about query response? Explain.
13. What are transactions? State the methods provided by PEAR DB for dealing with it.
14. What are sequences? How are they useful?
15. Explain the use of getListOf() method with suitable example.

Q.III Write the syntax and use of the following functions/methods.

1. pg_fetch_row()
2. pg_fetch_object()
3. pg_field_num()
4. fetchInto()
5. executeMultiple()
6. getCol()
7. getRow()
8. getAssoc()
9. tableInfo()
10. nextId()
11. createSequence()

***

6... AJAX and XML

Objectives...

- To learn the AJAX basic concepts.
- To learn how validation is performed using AJAX.
- To understand XML data handling using AJAX and PHP.
- To understand concept of XML.
- To study document structure of XML.
- To understand how PHP works with XML.
- To learn about XML Parser.
- To study Document object model and XML extension.

6.1 INTRODUCTION AJAX

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating dynamic, faster and interactive web based applications using XML, HTML, CSS, and JavaScript.
- AJAX uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic web page content display.
- XML is commonly used format for receiving server data. It also supports plain text format.
- AJAX depends on following technologies to create interactive and dynamic webpages:
  - **JavaScript:** JavaScript function is called when an event occurs in a page.
  - **DOM:** It supports use of API for accessing and manipulating structured documents. It also represents the structure of XML and HTML documents.
  - **CSS:** It allows you to separate presentation style from the content and may be changed programmatically by JavaScript.
  - **XMLHttpRequest:** It is a JavaScript object that performs asynchronous interaction with the server.
- All the available browsers cannot support AJAX . Here is a list of major browsers that support AJAX .
  - Mozilla Firefox 1.0 and above.
  - Netscape version 7.1 and above.
  - Apple Safari 1.2 and above.
  - Microsoft Internet Explorer 5 and above.
  - Konqueror.
  - Opera 7.6 and above.

Synchronous Vs Asynchronous Communication:

- In traditional communication between web client and web server, you fill out form data through web page, hit submit button and get directed to a new page with response received from the server. This type of communication between web client and web server is called as **Synchronous Communication.**
- With AJAX, when you hit submit button, JavaScript will make a request to the server, interpret the results, and update part of the current web page without reloading the complete web page. The user would never know that anything was even transmitted to the server. This type of communication between web client and web server is called as **Asynchronous Communication.**

Features of Asynchronous Communication:

- A user can continue to use the web application while the client program requests information from the server in the background.
- Always button click is not required, mouse events can also be used to send request to the server.
- Data-driven as opposed to page-driven.

Advantage of AJAX :

- It is independent of server technology.
- It uses JavaScript. So it can be used for all browser types.
- Using AJAX you can develop faster and more interactive web applications.
- Ajax based application use less server bandwidth, because there is no need to reload complete page. Partially current page web can be updated.

AJAX can't be used when:

- Page need to show in a search engine.
- Browser does not support JavaScript.
- User wants to create secure application.

Disadvantages of AJAX:

- AJAX is dependent on JavaScript. If JavaScript is disabled then, AJAX also will not useful.
- AJAX can have problems in Search engines as it uses JavaScript.
- Mostly in AJAX, GET methods is used, so there will be some security issues.
- Debugging is difficult.
- Increases size of the requests.
- The problem with browser back button, when AJAX is used.

6.2 UNDERSTANDING JAVA SCRIPTS FOR AJAX

- The user defined function written in JavaScript is called when an event occurs in a page.

- Use following steps for writing function definition in JavaScript to implement AJAX :
  1. Create an XMLHttpRequest Object.
  2. Send a request to a Web Server using GET or POST method.
  3. Display response received from Web server.

1. Create a XMLHttpRequest Object:

- All recent browsers versions have a built-in XMLHttpRequest object.
- **Syntax:** variable = new XMLHttpRequest();
- Old versions of Internet Explorer (IE5 and IE6) use an ActiveX Object.
- **Syntax:** variable = new ActiveXObject("Microsoft.XMLHTTP");
- While creating XMLHttpRequest object, check for browser's support as follows:

```
var xmlhttp;
if (window.XMLHttpRequest)
{
    // for recent browser' versions
    xmlhttp = new XMLHttpRequest();
}
else
{
    //for old IE5 and IE6
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

2. Send a request to a Web Server using GET or POST method:

- For sending request to a web server use open() and send() methods of XMLHttpRequest Object.
- **Syntax:** xmlhttpobject.open(method, file_name/url, boolean_value)

Where,

method can be GET or POST.

file_name/url which actually handles the request and process it.

boolean_value can be True for asynchronous request or False for synchronous request.

- The open() method prepares XMLHttpRequest object and send() method is used to send the request to web server.
- **GET Method:**
  - A GET method is used to send a request to a web server.
  - **Example:**

```
xmlhttp.open("GET", "demo.php", true);
xmlhttp.send();
```

- If you want to send information or form data with the GET method, add it to the URL:
- **Example**:

```
xmlhttp.open("GET", "demo.php?gamename=cricket", true);
xmlhttp.send();
```

- If you want to send two or more form elements then use ampersand (&) sign between the two elements.
- **Example**:

```
xmlhttp.open("GET", "demo.php?gamename=Cricket&player=Sachin",
true);
xmlhttp.send();
```

- **POST Method:**
  - A POST method is used to send a request to a web server.
  - **Example**:

```
xmlhttp.open("POST", "demo.php", true);
xmlhttp.send();
```

  - If you want to send information or form data, add an HTTP header with `setRequestHeader()` and specify the data you want to send in the send() method:
  - **Example**:

```
xmlhttp.open("POST", "demo.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xmlhttp.send("gamename=Cricket&player=Sachin");
```

  - The `setRequestHeader()` method **syntax**: `setRequestHeader(header, value);`
- **The File name or url - A File on a Server:**
  - This parameter of the `open()` method, is an address to a file on a server which is to be invoked.
  - Example: `xmlhttp.open("GET", "demo.php", true);`
  - The file can be any file like .txt and .xml. You can also specify .php file which will perform processing at server side before sending response to the browser.
- **Boolean Value – true:**
  - The `open()` method contains last parameter as true indicates request send to the server is asynchronous.
  - When using Boolean_value = true, specify a function to execute when the response is ready in the `onreadystatechange` event:
  - Example:

```
xmlhttp.onreadystatechange = function()
{
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
    {
        document.getElementById("demo").innerHTML = xmlhttp.responseText;
    }
};
xmlhttp.open("GET", "student.txt", true);
xmlhttp.send();
```

- **Boolean Value – false:**
  - The open() method contains last parameter as false indicates request send to the server is synchronous. JavaScript will not continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop. No further operation can be performed. No need to write onreadystatechange function.
  - Example:

```
xmlhttp.open("GET", "student.txt", false);
xmlhttp.send();
document.getElementById("demo").innerHTML = xmlhttp.responseText;
```

3. **Display response received from Web server:**
   - To get the response from the web server, use the responseText or responseXML property of the XMLHttpRequest object.
   - **responseText Property:** To get the response from the server as a string, use responseText property of XMLHttpRequest object.

     Example:

```
document.getElementById("demo").innerHTML = xmlhttp.responseText;
```

   - **responseXML Property:** To get the response from the server as XML file, use responseXML property of XMLHttpRequest object.

     Example:

```
document.getElementById("demo").innerHTML = xmlhttp.responseHTML;
```

- **The onreadystatechange event:**
  - When a request is sent to the server, you can perform some actions based on the response.
  - The onreadystatechange event is triggered every time the readyState changes.
  - The readyState property holds the status of the XMLHttpRequest object.
  - The onreadystatechange property stores a callback function to be invoked automatically each time the readyState property changes.
  - The readyState property holds current status of XMLHttpRequest object. The possible values are as follows:

Value	Description
0	Request not initialized
1	Server connection established
2	Request received
3	Processing request
4	Request finished and response is ready

  - The onreadystatechange event will be triggered five times, 0-4 corresponding to each value of readyState property.

The status property has following possible values:

0. **status:** Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
0. **statusText:** Returns the status as a string (e.g., "Not Found" or "OK").

Value	Description
200	OK
404	Page not found

6.3 AJAX WEB APPLICATION MODEL

- The web browser accepts input from the user and JavaScript XMLHttpRequest object's open() and send() methods sends the request to the server depending upon the event handled in the JavaScript.
- For sending request, either POST or GET method is used depending upon whether data is sensitive or not. With POST method, form data is sent through sent() function whereas with GET method, open() function sends form data to the server.
- At web server, processing can be done using PHP script to create response for the request. During processing, there might be need to fetch required information from database, XML file or text file etc.
- The result of the processing is sent as HTML or XML file which will be updated on the current web page. The working of AJAX web application model is represented in the following diagram.

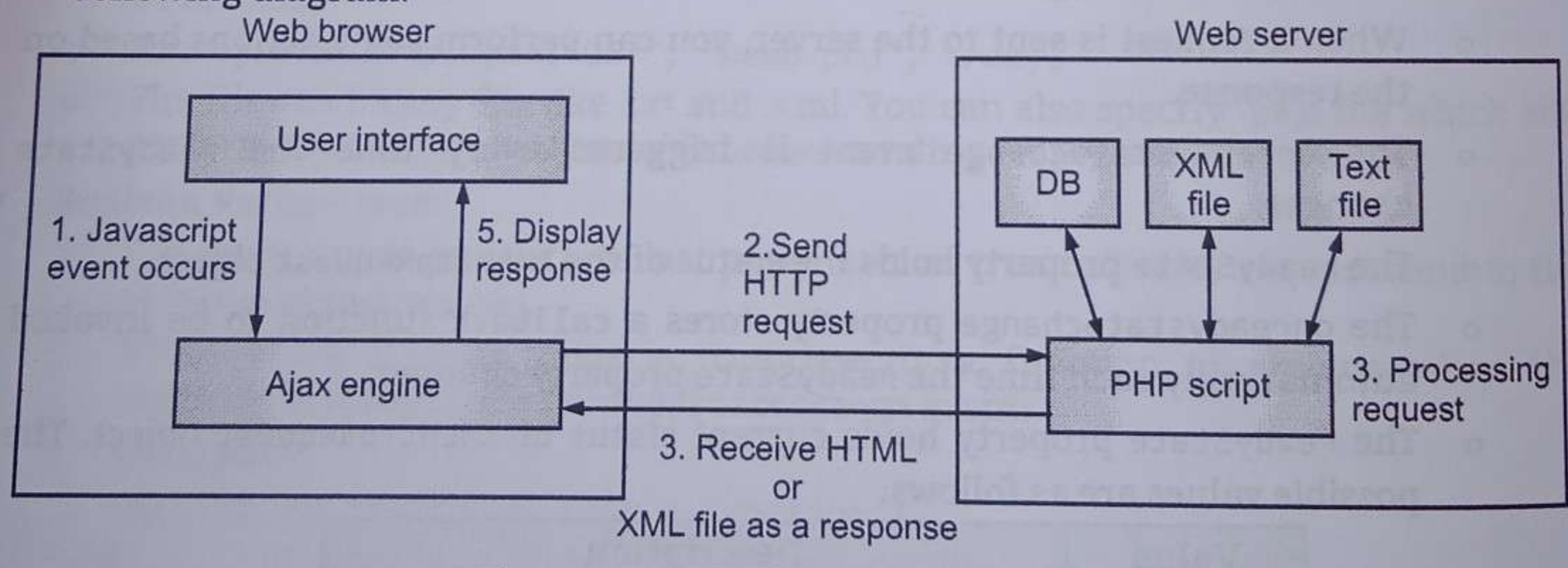


Fig. 4.1: AJAX Web Application Model

- Here, communication between web browser and web server can be synchronous or asynchronous. In synchronous communication, user has to wait for the response from the server till that no other task can be performed. This may include waiting time for the user.
- In asynchronous communication, user can continue to use the web application while the client program requests information from the server in the background.

6.4 AJAX - PHP FRAMEWORK

- A AJAX PHP framework is able to deal with database, searching of data, and build pages or parts of page and return the response such as HTML file or XML file to the `XMLHttpRequest` object.
- Using AJAX, you can pass name of php file along with form data as a second argument to `XMLHttpRequest` object's `open()` method to perform processing at server side. The server uses PHP file for processing the request. The response sent by the server will be displayed on the current web page, so it is possible to partially update the web page. This reduces the need to do a page refresh or full page reload for every user interaction.
- The following are AJAX framework which are used for creating web applications with a dynamic link between the client and the server:
  - Quicknet is an AJAX framework that provides secure data transmission, uses PHP on the server side.
  - SAJAX PHP framework with a lot of functions, easy to integrate functions yourself.
  - XAJAX uses JSON or XML format, on the server side.

Program 6.1: Write AJAX program to read a text file and print the contents of the file when the user clicks on the Print button.

file.html:

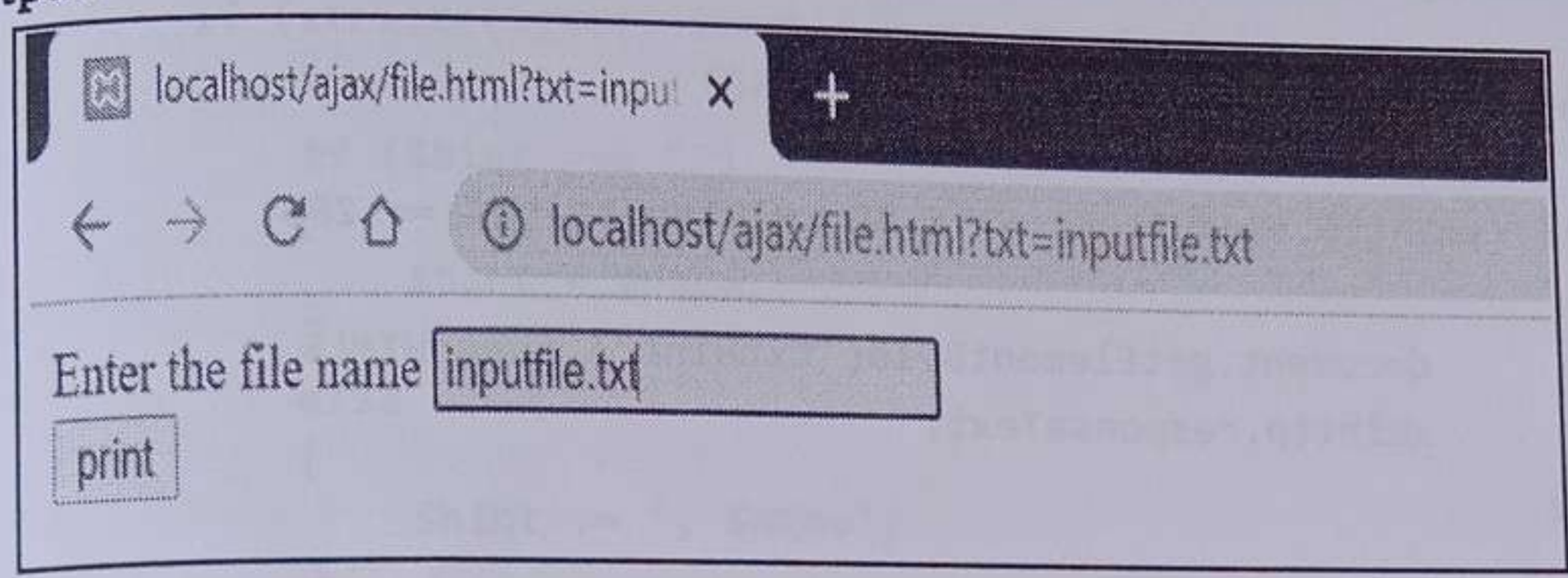
```
<html>
<head>
<script>
function displayFile()
{
    str=f1.txt.value;
    if(str=="" || str==null)
    {
        alert("Enter file name");
    }
    else
    {
        var xmlhttp;
        if(window.XMLHttpRequest)
          xmlhttp= new XMLHttpRequest();
```

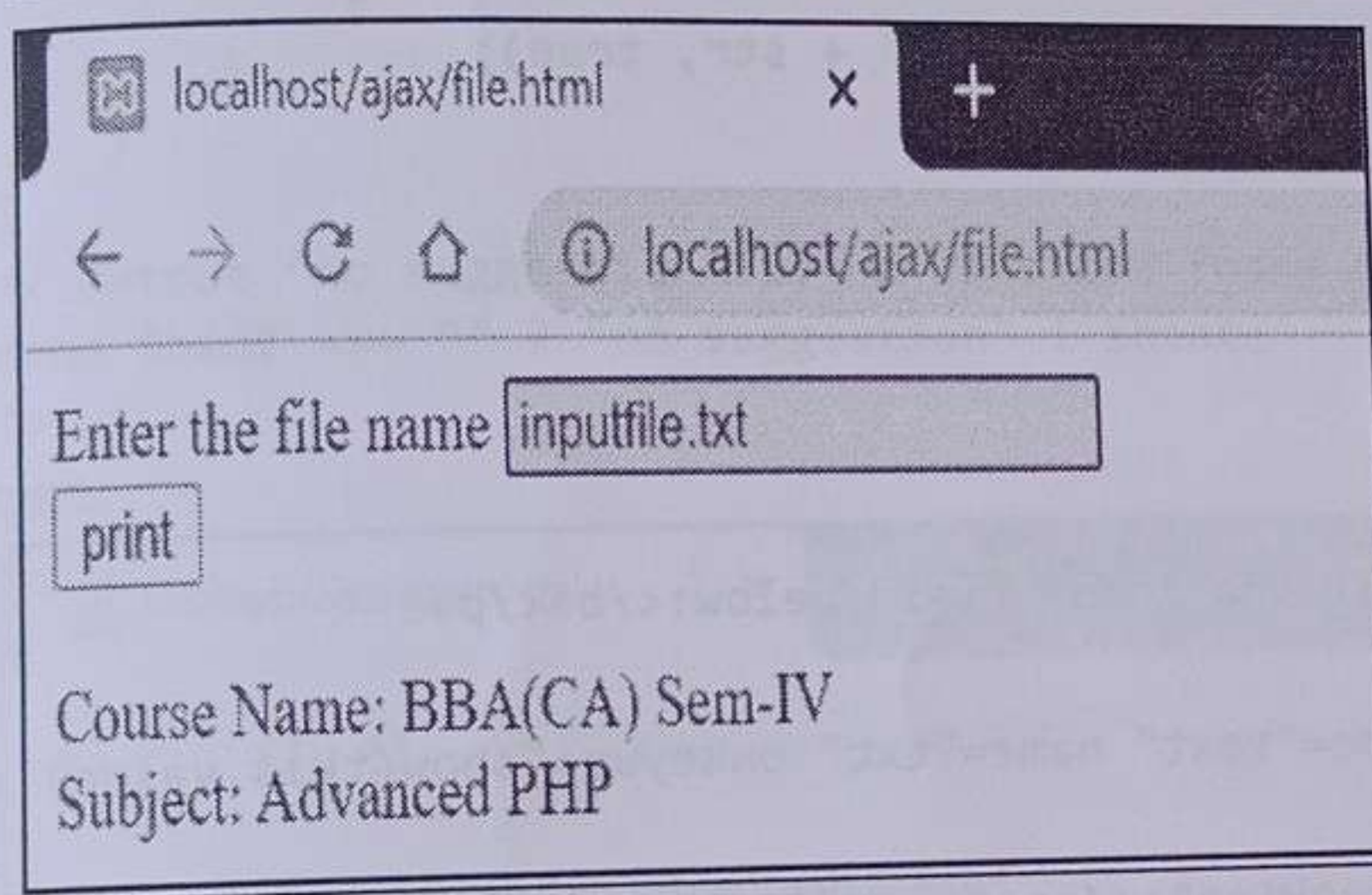
```
        else
          xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        xmlhttp.open("GET","file.php?txt="+str,false);
        xmlhttp.send();
        document.getElementById("result").innerHTML=xmlhttp.responseText;
    }
}
</script>
</head>
<body>
    <form name="f1" method="GET">
    Enter the file name
    <input type="text" name="txt" value=""/><br>
    <input type="button" value="print" onclick="displayFile()"/>
    <div id="result"></div>
    </form>
</body>
</html>
```

file.php:

```
<?php
$file1=$_GET['txt'];
if($fp=fopen($file1,"r"))
{
    while(($line=fgets($fp,80))!=false)
    {
        echo"<br>".$line;
    }
}
else
{
    echo"File does not exist.";
}
?>
```

Output:





Program 6.2: Write an AJAX program to display list of games stored in an array corresponding to search string given by user in the HTML form.

game.html:

```
<html>
<head>
<script>
function show(str)
{
   if (str.length == 0)
   {
      document.getElementById("txtHint").innerHTML = "";
      return;
   }
   else
   {
      var xmlhttp;
      if(window.XMLHttpRequest)
         xmlhttp= new XMLHttpRequest();
      else
```

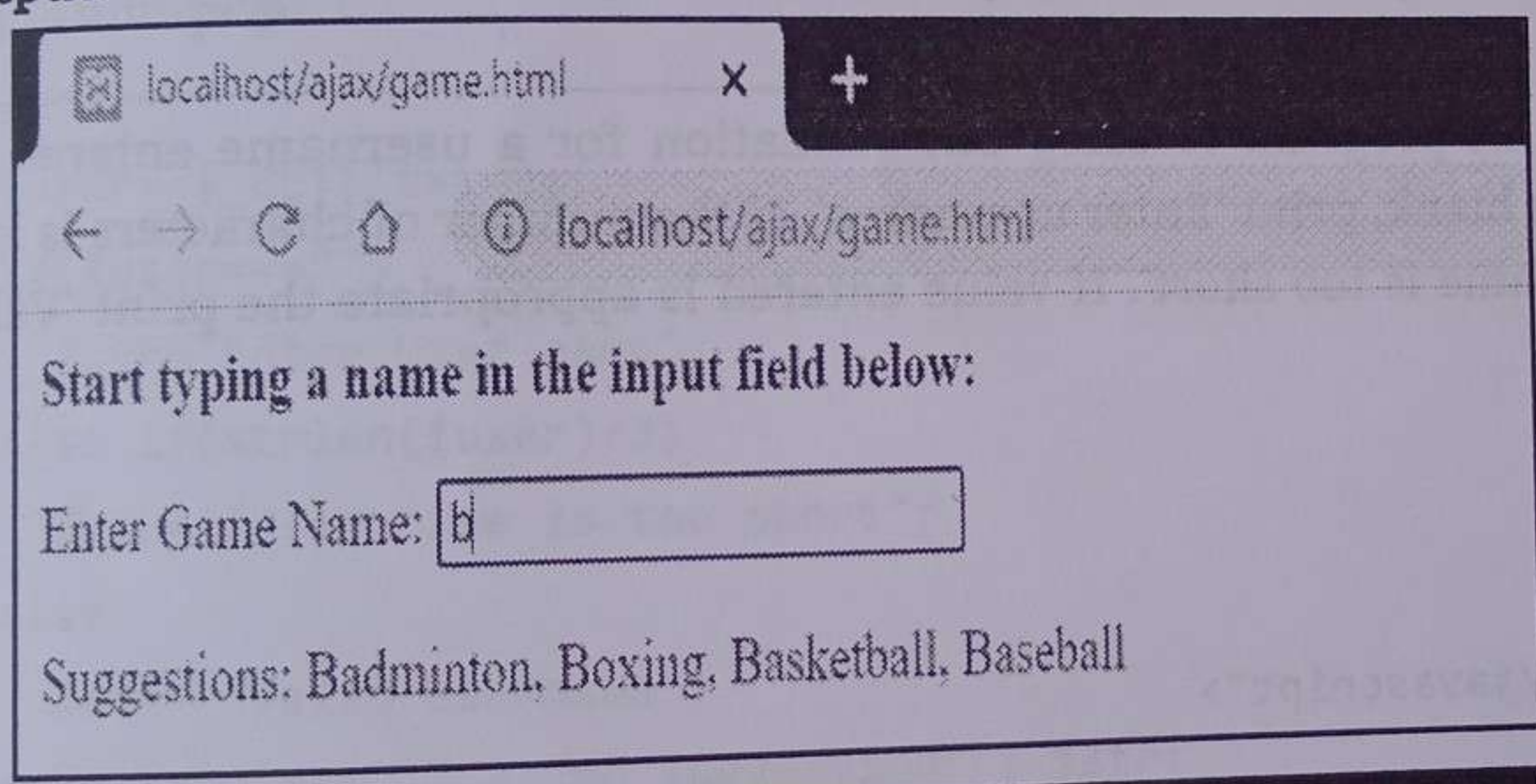
```
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

    xmlhttp.onreadystatechange = function()
    {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
        {
            document.getElementById("txtHint").innerHTML=
            xmlhttp.responseText;
        }
        };
        xmlhttp.open("GET", "game.php?txt=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form>
Enter Game Name: <input type="text" name="txt" onkeyup="show(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

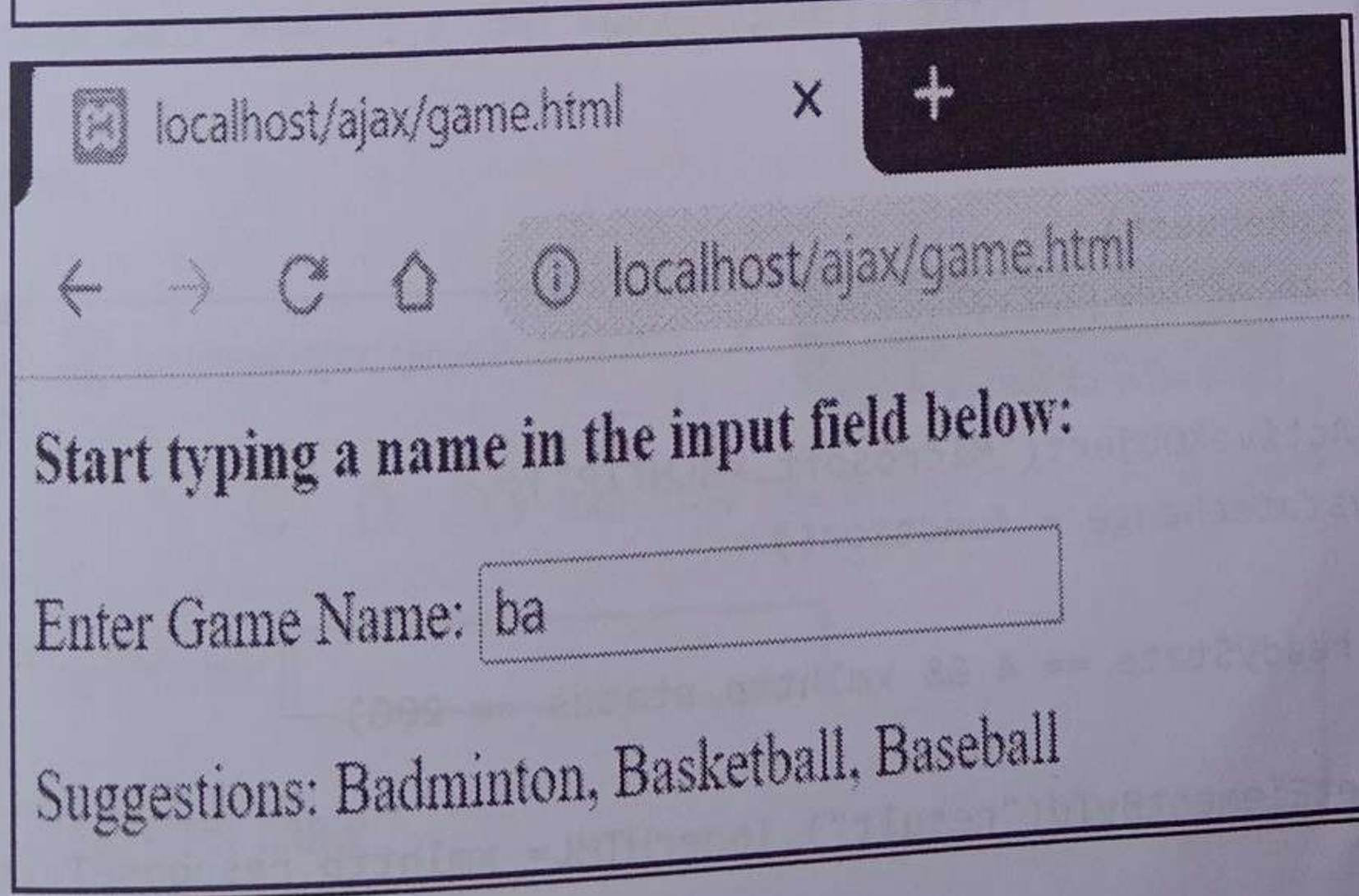
game.php:

```
<?php
// Array with game names
$a = array("Cricket", "Archery", "Badminton","Boxing","Curling", "Tennis",
"Skateboarding", "Surfing", "Hockey", "Yoga", "Gymnastics", "Karate",
"Weightlifting", "Volleyball","Wrestling", "Basketball",
"Baseball","Cycling", "Running", "Climbing");
// get the txt parameter from URL
$game = $_REQUEST["txt"];
$hint = "";
// lookup all hints from array if $game is not empty
if ($game !== "")
{
    $len=strlen($game);
    foreach($a as $name)
```

```
    {
        if (stristr($game, substr($name, 0, $len)))
        {
            if ($hint === "")
            {
                $hint = $name;
            }
            else
            {
                $hint .= ", $name";
            }
        }
    }
}
// Output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

Output:





- **1st Output Screen:** The character 'b' is entered, so all possible strings which are starting with 'b' has been displayed
- **2nd Output Screen:** Now characters 'ba' are entered, so all possible strings which are starting with 'ba' has been displayed.
- When search characters are entered by user, these characters are sent to the server. In PHP file input characters are matched from the beginning of each string from the array which stores game names and all possible matching strings are displayed.

6.5 PERFORMING AJAX VALIDATION

- The form validation is one of the important tasks for web based applications. JavaScript `XMLHttpRequest` object along with PHP can be used for form data validation.
- When user enters HTML form data, it is send to **.php** file using AJAX for validation purpose. Once the validation is done, PHP will send response to the browser and it will be displayed on the same web page.
- Here, browser can send synchronous or asynchronous request to the server for form data validation.

Program 6.3: Write AJAX program to carry out validation for a username entered in textbox. If the textbox is blank, print 'Enter username'. If the number of characters is less than three, print Username is too short'. If value entered is appropriate the print 'Valid username'.

validation.html:

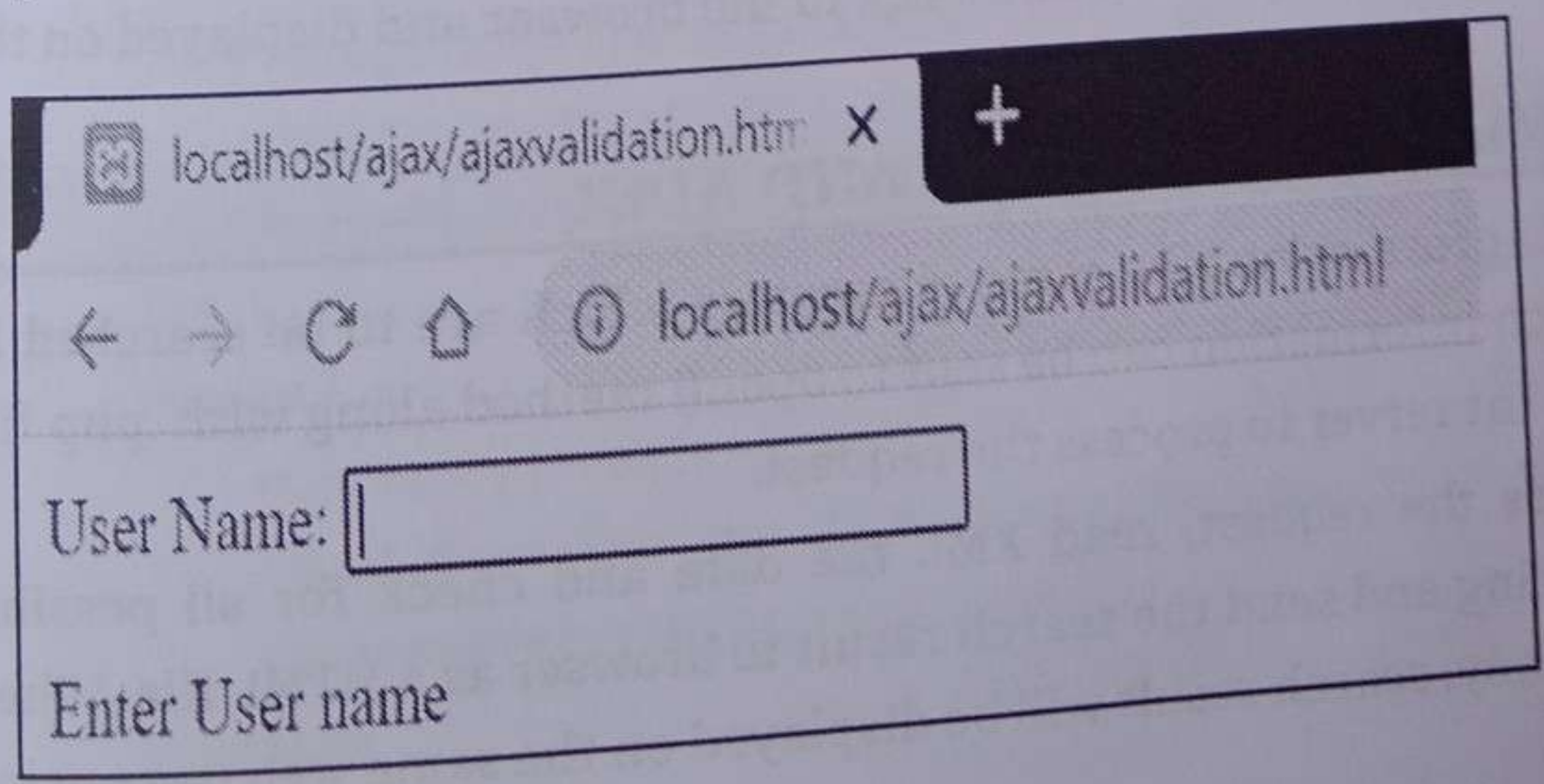
```
<html>
<head>
<script type="text/javascript">
function show(str)
{
    var xmlhttp;
    if(window.XMLHttpRequest)
      xmlhttp= new XMLHttpRequest();
    else
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    xmlhttp.onreadystatechange = function()
    {
      if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
      {
        document.getElementById("result").innerHTML= xmlhttp.responseText;
      }
```

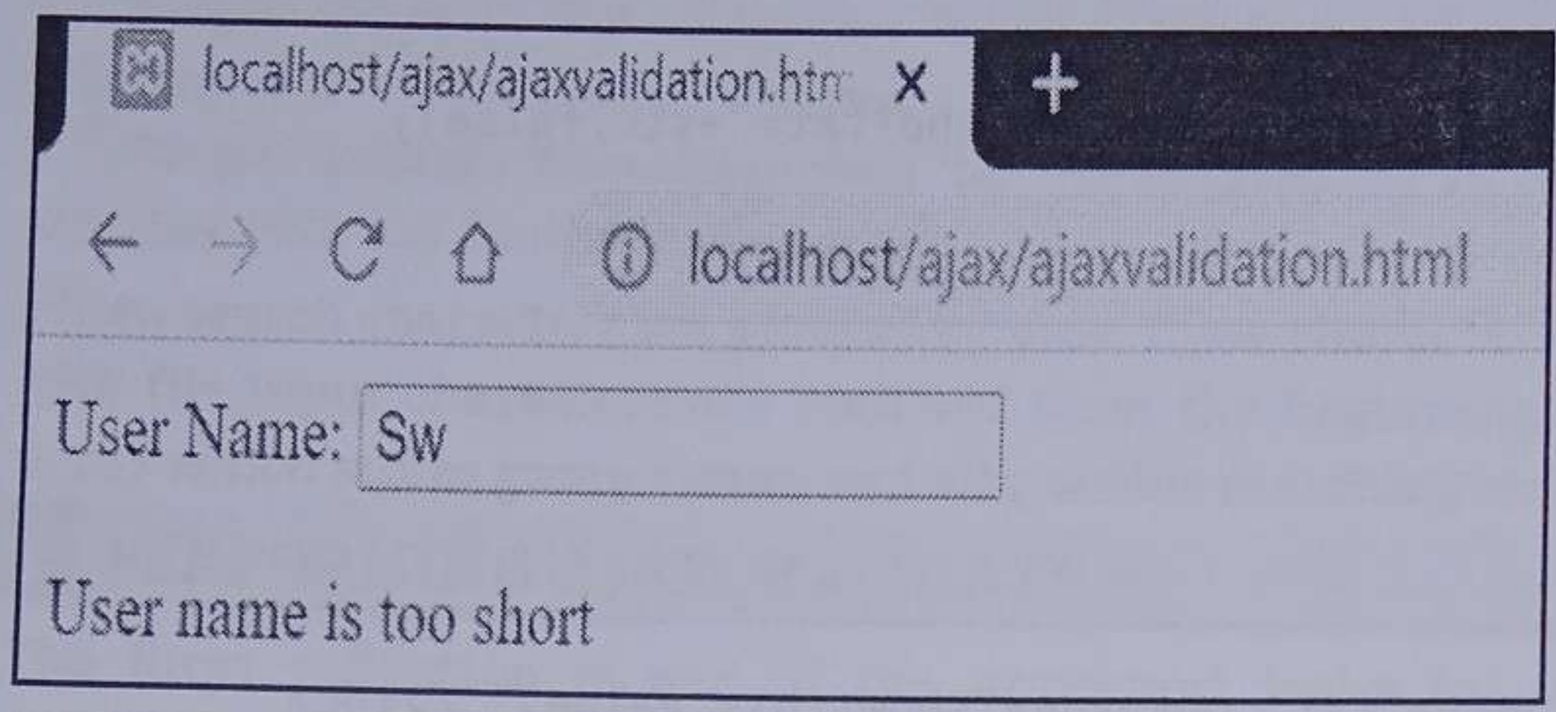
```
    };
    xmlhttp.open("GET","AJAX validation.php?txt="+str,false);
    xmlhttp.send();
}
</script>
<body>
<form>
User Name:
<input type="text" name="txt" onkeyup="show(this.value)" /><br>
</form>
<div id='result'></div>
</body>
</html>
```

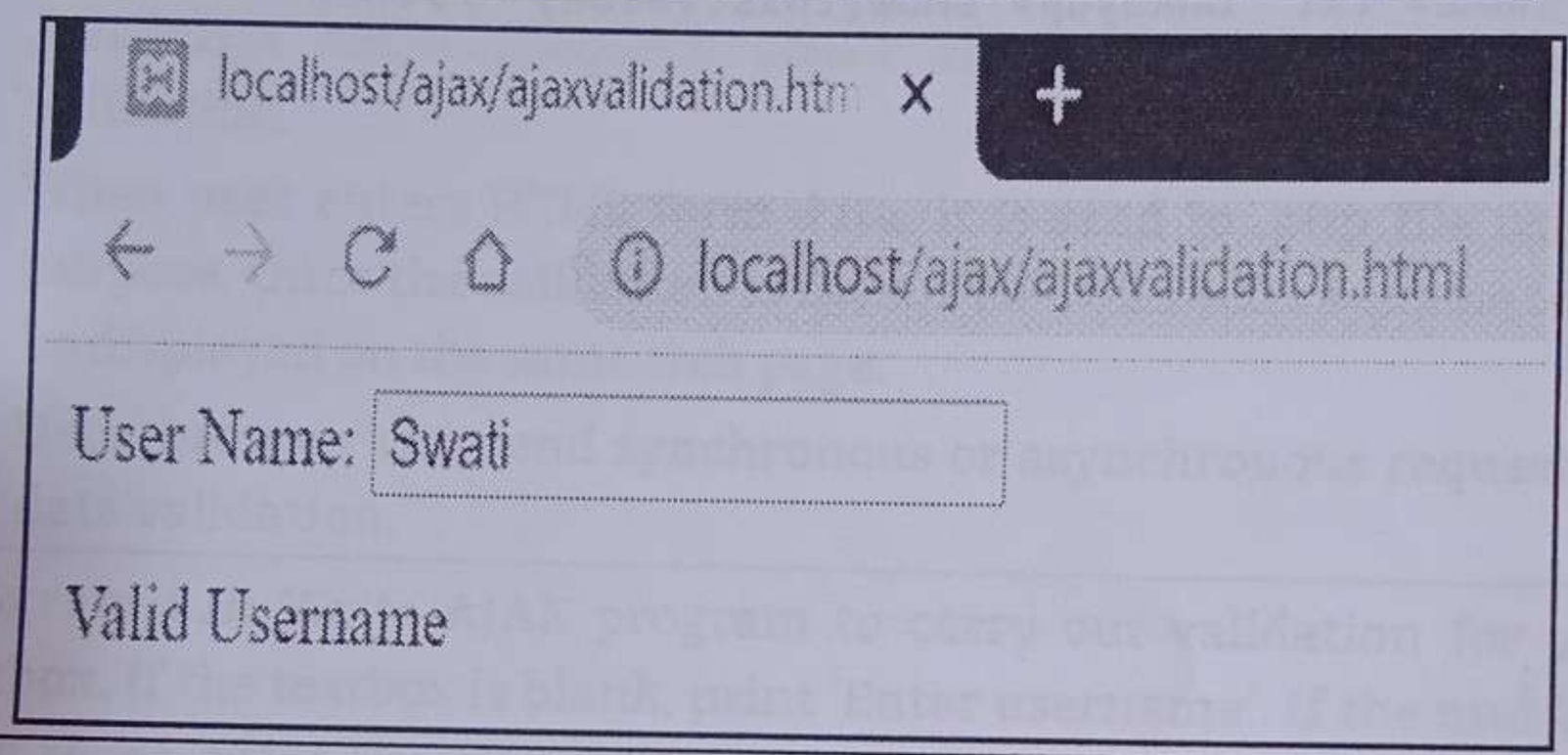
validation.php:

```
<?php
$user= $_GET['txt'];
if($user==="")
  $str="Enter User name";
else if(strlen($user)<3)
  $str= "User name is too short";
else
  $str= "Valid Username";
echo $str === "" ? "no suggestion" : $str;
?>
```

Output:







- **1st Output Screen:** When user name is empty, message sent by the server is "Enter User name".
- **2nd Output Screen:** When length of user name is less than 3, message sent by the server is "User name is too short".
- **3rd Output Screen:** When length of user name is greater than 3, message sent by the server is "Valid User name".
- The AJAX `validation.html` file accepts username and sends it to server using JavaScript `XMLHttpRequest` object's open() and send() method. In `validation.php` file, validation is performed and response is sent back to the browser and displayed on the same web page.

6.6 HANDLING XML DATA USING PHP AND AJAX

- You can accept input information like some characters which are to be searched in the XML file. The search information can be sent to open() method along with .php file which will be executed at server to process the request.
- In .php file, to process the request, read XML file data and check for all possible matches for search string and send the search result to browser as a HTML file. When the response will be ready, search result will be displayed on the same web page using AJAX.

- The steps to write AJAX code to retrieve information from XML file using PHP:
  1. Accept search string from user.
  2. Send search string to open() method along with .php file name.
  3. In .php file, read XML file data.
  4. Find all possible matches for search string with XML file data.
  5. Display search result in PHP script to create HMTL response for the given request.
  6. This output will be sent to browser and will be displayed on same web page using AJAX.
- All above steps are illustrated in the following program.

Program 6.4: Write AJAX program to accept the movie name and print all possible suggestions for movie name from XML file. Also display movie details. Consider `movie.xml` file as follows:

movie.xml:

```
<?xml version="1.0"?>
<Movie_Store>
<Movie>
        <Category>Biography</Category>
        <MovieName>Dangal</MovieName>
        <ReleaseYear>2016</ReleaseYear>
    </Movie>
    <Movie>
        <Category>Action</Category>
        <MovieName>Tanhaji</MovieName>
        <ReleaseYear>2020</ReleaseYear>
    </Movie>
    <Movie>
        <Category>horror</Category>
        <MovieName>Bhoot</MovieName>
        <ReleaseYear>2020</ReleaseYear>
    </Movie>
    <Movie>
        <Category>Family</Category>
        <MovieName>The Sky Is Pink</MovieName>
        <ReleaseYear>2019</ReleaseYear>
    </Movie>
</Movie_Store>
```

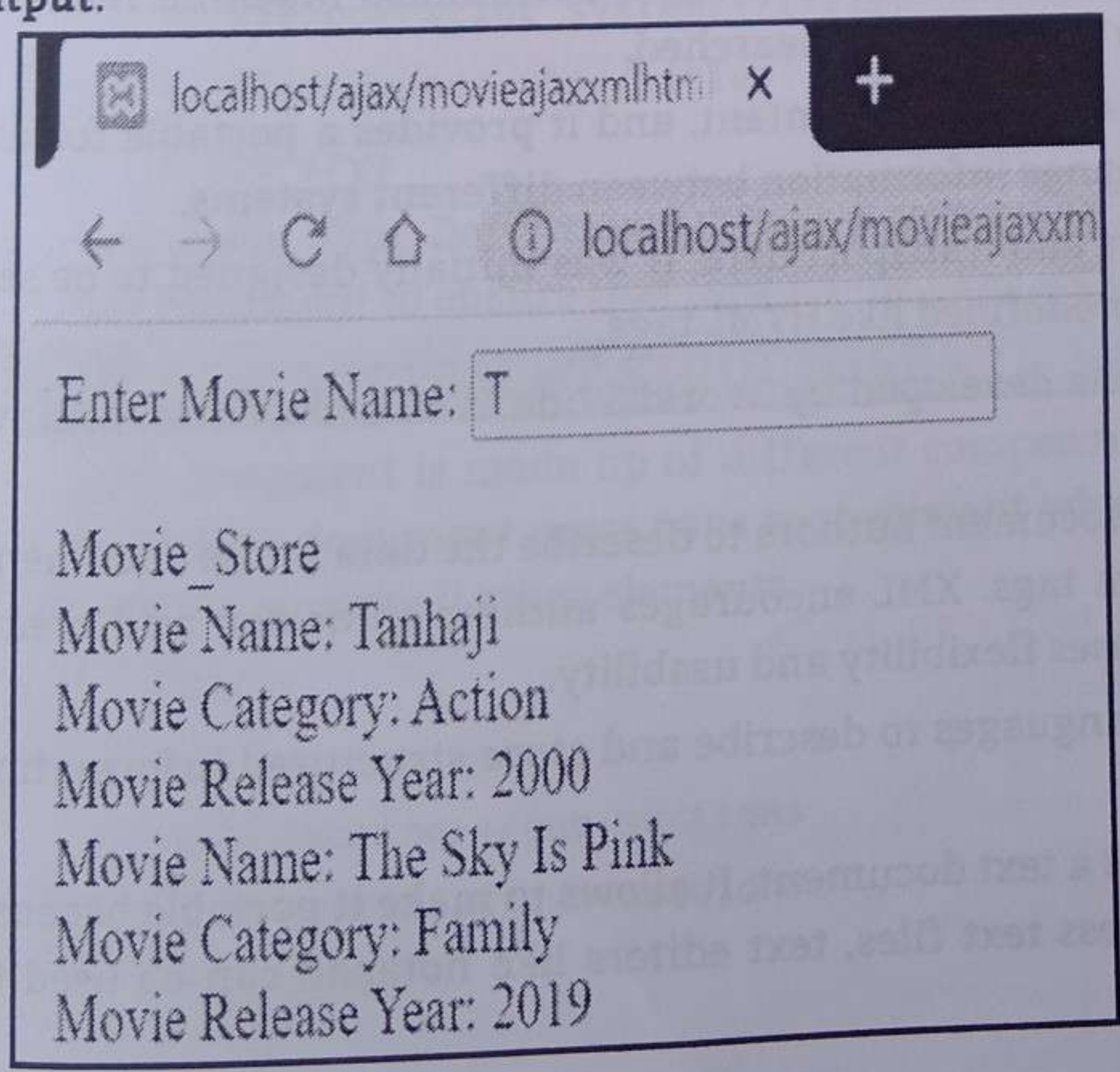
movie.html:

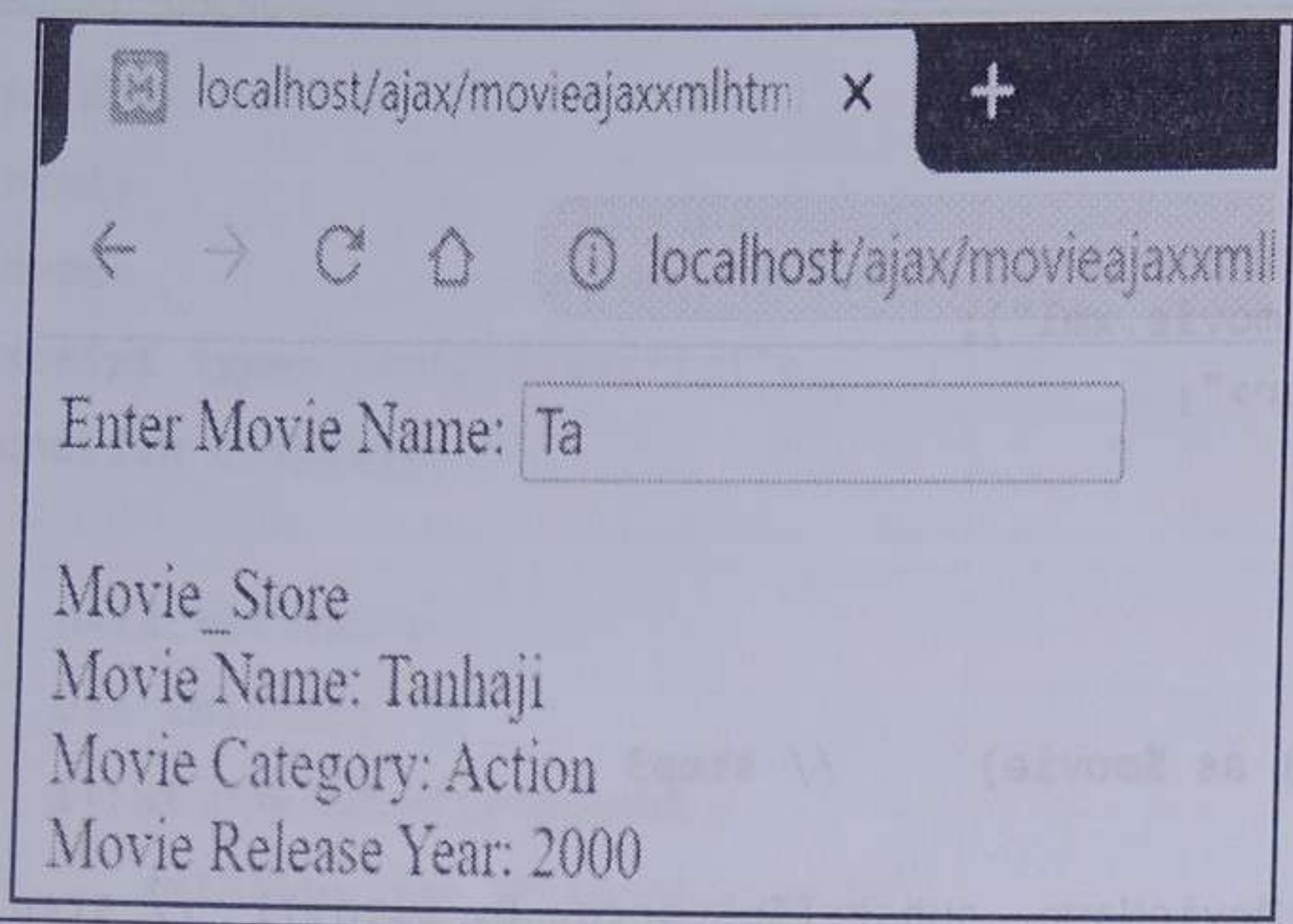
```
<html>
<head>
<script type="text/javascript">
function display()
{
    n=f1.txt.value;
    var xmlhttp;
    if(window.XMLHttpRequest)
        xmlhttp= new XMLHttpRequest();
    else
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    xmlhttp.onreadystatechange=function()
    {
      if(xmlhttp.readyState==4 && xmlhttp.status==200)
      {
          document.getElementById('result').innerHTML = xmlhttp.responseText;
          // step 6
      }
    };
    xmlhttp.open("GET","movie.php?txt="+n,true); // step2
    xmlhttp.send();
}
</script>
<body>
<form name="f1">
Enter Movie Name:
<input type="text" name="txt" onKeyUp="display()"/><br>  // step1
</form>
<div id='result'></div>
</body>
</html>
```

movie.php

```
<?php
$msearch=$_GET['txt'];
$xml=simplexml_load_file("movie.xml");
echo $xml->getName() . "<br>";
$hint = "";
$len=strlen($msearch);
if ($msearch !== "")
{
  foreach($xml->children() as $movie)     // step3
  {
      if (strstr($movie->MovieName, substr($msearch, 0, $len)))  // step 4
      {
          $hint .= " Movie Name: $movie->MovieName <br>
          Movie Category: $movie->Category <br>
          Movie Release Year: $movie->ReleaseYear<br>";
      }
  }
}
// Output "no suggestion" if no hint was found or output correct values

echo $hint === "" ? "no suggestion" : $hint;
?>
```

Output:





- **1st Output Screen:** When user types 'T' character then onkeyUp() event occurs. XMLHttpRequest object is sent to the server which executes .php file. The .php file reads XML file contents and 'T' is matched with the XML file data and matching result is sent to the browser.
- **2nd Output Screen:** When search characters changes every time, .php file be executed and displays new search result.

6.7 WHAT IS XML?

- XML is extensible Markup Language; it is a standardized data format. XML has similar tags like HTML but XML documents are parsed. XML is used as standard data format in various fields like publishing, engineering and medicine.
- It allows content authors to mark up their data with customized machine readable tags, so that data can be easily classified and searched.
- XML helps to enforce formal structure on content, and it provides a portable format that can be used to easily exchange information between different systems.
- It is basically designed to store and transport data. It was initially designed to be self descriptive. XML tags are not predefined like HTML tags.
- XML is a public standard it was developed by World Wide Web consortium and it's visible as open standard.
- It is a language that helps the document authors to describe the data in the document, by marking it up with custom tags. XML encourages authors to design and create their own tags, so that it increases flexibility and usability.
- It is one of the most popular languages to describe and store structured information on the web.
- XML data is physically stored as a text document. It allows to make it portable because all systems can read and process text files, text editors like notepad can be used to

create XML documents, Mozilla Firefox, internet explorer supports it and they can be used to read and display data.

- Most of the XML documents consist of elements like HTML tags, entities, and regular data.

```
<book isbn="19256245612">
<title>Advance PHP </title>
<authors>
<author>swatijadhav</author>
<author>gajanandeshmukh</author>
</authors>
</book>
```

- XML requires every tab to be closed. In XML, tags can be nested but can't be overlapped.
- XML also requires that the document begin with processing instructions that identifies the version of XML being used, for example: `<?xml version="1.0">`
- Another requirement of well-formed XML document is that there can be only one element at the top level of the file. Let's take an example,

```
<?xml version="1.0">
<library>
<title>programming in java</title>
<title>operating systems</title>
<title>advance php</title>
</library>
```

- Document Type Definition (DTD) and schema. DTD and schema are used to validate the document to ensure that they follow rules for their type of document.

6.8 XML DOCUMENT STRUCTURE

- XML document is made up of different components each serving a specific purpose. Each XML document must have root element which is the starting of XML document which contains all other elements.

```
<root>
<section>
<sub-section></sub-section>
<sub-section></sub-section>
<section>
```

```
<section>
<sub-section></sub-section>
<sub-section></sub-section>
<section>
</root>
```

- XML document must begin with a prolog that appears before the root element. It contains metadata about the element, like character encoding, document structure and style sheets.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- XML tags are case sensitive and they should begin with opening tag < > and must end with < / > tag.

  e.g. `<name></name>`

- An element in XML is formed by characters between the start tag and end tag.

The major portion of XML document includes the following:

0. XML Prolog:

- The Prolog contains optional information such as the XML version the document conforms to, information about the character encoding used to encode the contents of the document, and a reference to either a Document Type Definition (DTD) or XML Schema document which describes the grammar and vocabulary of the document.
- The XML Schema document is the more modern way to describe XML grammar and vocabulary.
- XML Schemas and DTDs are usually stored in external documents and the prolog can reference both XML Schemas and DTDs.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE maillist SYSTEM "maillist.dtd">
```

0. XML Body:

- The body contains a single top level element called the root element, which contains all other elements and other markup information.
- Following simple example explains the prolog and body, where addresses is root element and encloses entire body, of XML document.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE maillist SYSTEM "maillist.dtd">
<addresses>
   <address></address>
   <address></address>
   <address></address>
</addresses>
```

0. **DTD and XML Schema Documents:**

- DTD and XML Schema documents are rules that define the elements that can exist in a particular document or group of documents, and the relationships among the various elements.
- A DTD or XML Schema can be part of the content of an XML document or can be separate from it and referred to by the XML documents.
- Best practice calls for the DTD and XML Schema documents to be separate from the XML content for reuse and maintainability.
- Here is an example of a DTD:

```
<?xml encoding="US-ASCII"?>
<!-- DTD for a an XML document that stores Customer names and numbers-->
<!ELEMENT customer(name, cust-num)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT cust-num(#PCDATA)>
```

- An example of XML Schema:

```
<?xml version="1.0"?>
<!-- XML Schema document for a an XML document that stores Customer
Names and numbers-->
<xsd:schemaxmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="">
<xsd:element name="customer" minoccurs="0" maxoccurs="unbounded">
<xsd:complextype>
<xsd:sequence>
<xsd:element name="custnum" type="xsd:int"/>
<xsd:element name="Name" type="xsd:string"/
</xsd:sequence>
</xsd:complextype>
</xsd:element>
</xsd:schema>
```

0. **XML Elements:**

- Elements represent the logical components of documents. They can contain data or other elements. For example, a customer element can contain a number of column (field) elements and each column element can contain one data value.
- An element is composed of a start tag, data, and an end tag. Here is an example of an element:
  `<cust-name>suresh</cust-name>`, here we have starting tag, data and ending tag.
- Let's take look at example containing other element's:

```
<phone>
    <entry>
        <name>suresh</name>
        <extension>120</extension>
    </entry>
</phone>
```

- In this example, name and extension are child elements of entry, which is a child element of phone. Similarly, phone is a parent element of entry, which is a parent element of name and extension.
- The top level element, which is parent of all elements, is referred as root element.
- A child element that can't have its own child elements as defined by DTD or XML schema is referred as leaf element.
- A typical XML elements that describe hierarchy uses following terms:
  - Root element
  - Parent element
  - Child element
  - Leaf element

0. **XML Attributes:**

- Elements can have additional information called attributes attached to them. Attributes describe properties of elements. Here is an example of an element with an attribute, emp_num:

```
<employee id="1">
    <name>suresh</name>
</employee>
```

0. **XML empty elements:**

- An empty element is an element that is complete by itself; it never contains other elements. Rather than being composed of a start tag, data, and an end tag, the empty element is a combined start and end tag.
- Typically, an empty element is used as a flag of some kind or perhaps as an element that contains its information in its attributes. In this example, extension has been converted to an empty tag:

```
<phone>
    <entry>
        <name>suresh</name>
        <extension number="120"/>
    </entry>
</phone>
```

0. **Comments:**

- Comments can be placed anywhere in a document and they are not considered to be a part of textual content of XML document.
- **Syntax:** `<!-- comment tag -->`

6.9 PHP AND XML

- Native XML support in PHP4 was limited to some specific technologies; new version of PHP5 has introduced several new features and enhancements to PHP language.

Core XML Extensions:

3. **Tree based parsers:**

- Tree based parsers allows you to construct or load existing XML documents so you can navigate or modify them. To achieve this, entire XML document is created or loaded into memory as tree. But the entire document must reside in memory.

3. **DOM Extension:**

- The DOM extension is the PHP5 replacement of `domxml`, which is now supported only under PHP4.
- DOM extension was created to address shortcomings of `domxml` while adhering to W3C DOM specifications; DOM extension has large and complex API.
- The DOM extension allows you to access all node types, it also allows you to create and modify complex documents, and gives you navigation and functionality.

3. **SimpleXML Extension:**

- Using new functionality offered by PHP5, `SimpleXML` provides an extremely simple and lightweight tool to manipulate XML documents. Compared to the DOM extension, `SimpleXML` has easy to learn API because you can view the document as tree of objects, where objects are synonymous with element nodes.
- Accessing child elements is simple, you can access attributes similarly just like how access an array. To some extent, `SimpleXML` allows content editing.
- To create a `SimpleXML` object from an XML document stored in a string, pass the string to `simplexml_load_string()`. It returns a `SimpleXML` object. `simplexml_load_file()` this function is used to load external XML file.

Program 6.5: PHP program to read the data from XML file using PHP code.

Employee.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees dept = "civil">
    <record emp_id = "101">
        <name>sureshshinde</name>
        <position>CEO</position>
    </record>
    <record emp_id = "102">
        <name>rameshdeshpande</name>
        <position>Marketing Manager</position>
    </record>
    <record emp_id = "103">
        <name>sachinjoshi</name>
        <position>Production Manager</position>
    </record>
</employees>
```

```
<?php
$xml = simplexml_load_file('Employee.xml');
echo '<h2>Employees Information</h2>';
$list = $xml->record;
for ($i = 0; $i < count($list); $i++)
```

```
{
    echo 'Emp_id: ' . $list[$i]->attributes()->emp_id . '<br>';
    echo 'Name: ' . $list[$i]->name . '<br>';
    echo 'Position: ' . $list[$i]->position . '<br><br>';
}
?>
```

Output:

Employees Information

Emp_id: 101
Name: suresh shinde
Position: CEO

Emp_id: 102
Name: ramesh deshpande
Position: Marketing Manager

Emp_id: 103
Name: sachin joshi
Position: Production Manager

- Here we have used `simplexml_load_file()` function, to load the XML file and assigns the content to the array variable $xml.
- "`$list = $xml->record;`" gets the contents of the record node.
- "`for ($i = 0; $i < count($list); $i++)`" is the for loop that reads the numeric array and outputs the results.
- "`$list[$i]->attributes()->emp_id;`" reads the `emp_id` attribute of the element.
- "`$list[$i]->name;`" reads the value of the name child element.
- "`$list[$i]->position;`" reads the value of the position child element.

Program 6.6: PHP program for `simplexml_load_string()` function, it returns simpleXML object.

```
<html>
 <body>
 <?php
 $note=<<<XML
 <note>
 <to>Abhay Deshmukh</to>
 <from>Gajanan Deshmukh</from>
 <heading>Greetings of the day</heading>
 <body>Hello How r u? </body>
```

```
</note>
XML;
$xml=simplexml_load_string($note);
print_r($xml);
?>
</body>
</html>
```

Output:

SimpleXML Element Object ([to] => Abhay Deshmukh [from] => Gajanan Deshmukh [heading] => Greetings of the day [body] => Hello How r u?)

6.9.1 Generating XML Document

- `SimpleXML` is an better option for parsing existing XML Documents, but you can't use to create new documents.
- The simplest way to generate an XML document is to build a PHP array whose structure mirrors that of the XML document and then to iterate through the array, printing each element with appropriate formatting.

Program 6.7: PHP program for generating XML Document.

```
<?php
$student = array('Student_Name'=> 'Santosh Mishra',
'Student_class'=> 'SYBBACA',
'Subject_Name' => 'Advance PHP');
print "<student>\n";
foreach ($student as $element => $content) {
print " <$element>";
print htmlentities($content);
print "</$element>\n";
}
print "</student>";
?>
```

Output:

Santosh Mishra SYBBACA Advance PHP.

6.10 XML PARSER

- XML parser is software library or package that provides interface for client applications to work with XML documents.
- XML parser is an event based parser, it works like an event handler. XML parser is designed to read XML and create way of programs to use XML. It validates documents and check the document is well formed.

- Typically XML document is processed by software known as XML parser. It reads XML documents using one of the two approaches, the simple API for XML (SAX) approach or the document object model approach.
- A SAX parser works by traversing XML document sequentially, from starting to end of document, and calling specific user defined functions as it encounters different types of XML constructs.
- It does not create any internal structure. Clients does not know what methods to call, they just override the methods of API and place his own code inside the method.
- PHP's XML parser is based on Expat C Library which allows you to parse but not to validate XML Documents.
- It means that you can check which XML tags are present but you can't validate whether they are in the right structure.

PHP XML parser extension includes several functions to access XML elements:

8. `xml_parser_create()`: It is used to create XML parser handler.
8. `xml_parser_create_ns()`: This function creates parser handler with namespace support.
8. `xml_parse()`: It is used to parse XML document.
8. `xml_parse_into_struct()`: It is used to covert XML nodes into array.
8. `xml_set_element_handler()`: To set start and end element handlers for XML parser. There are more setters to set variety of handlers.
8. `xml_get_current_line_number()`, `xml_get_current_column_number()`: To get current column line number and column number respectively.
8. `xml_parser_free()`:To cancel parser handler reference if it is not required.
8. `Handler_functions`: In XML Parser extension, handler functions are defined to invoke on a particular event. For example, start and end element handlers of this parser are invoked on element start and end respectively. For XML parser a start element handler must contain parameters like parser handler, element name, and its attribute array. End element handler must contain parser handler and element name. After definition, we need to set these element handlers to XML parser by using `xml_parser_element_handler()`.

6.10.1 XML DOM Parser

- The DOM extension allows you to access all node types, create and modify complex documents, and gives you advanced navigation and functionality.
- An advantage to this extension, if you are coming from another language that incorporates a DOM compliant parser, is that the API should already be familiar to you and easy to begin using under PHP.
- This parser makes it possible to process XML documents in PHP.

Program 6.8: PHP program to demonstrate DOM Document.

employee.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees dept = "civil">
<record emp_id = "101">
<name>suresh shinde</name>
<position>CEO</position>
<salary>50000</salary>
</record>
<record emp_id = "102">
<name>ramesh deshpande</name>
<position>Marketing Manager</position>
<salary>20000</salary>
</record>
<record emp_id = "103">
<name>sachin joshi</name>
<position>Production Manager</position>
<salary>40000</salary>
</record>
</employees>
```

employee.php

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("employee.xml");
print $xmlDoc->saveXML();
?>
```

Output:

suresh shinde CEO 50000 ramesh deshpande Marketing Manager 20000 sachin joshi Production Manager 40000

- The above example creates `DOMDocument` object and loads the XML from `employee.xml` into it.
- The `saveXML()` function puts the internal XML document into a string, so we can output it.
- Looping through XML: Now we want to initialize XML parser, load the XML document, and iterate though the all elements of employee.

Program 6.9: PHP code to iterate though `employee.xml`.

employee.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees dept = "civil">
<record emp_id = "101">
<name>suresh shinde</name>
<position>CEO</position>
<salary>50000</salary>
</record>
<record emp_id = "102">
<name>ramesh deshpande</name>
<position>Marketing Manager</position>
<salary>20000</salary>
</record>
<record emp_id = "103">
<name>sachin joshi</name>
<position>Production Manager</position>
<salary>40000</salary>
</record>
</employees>
```

Employee.php

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("employee.xml");
$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item)
{
   print $item->nodeName . " = " . $item->nodeValue . "<br>";
}
?>
```

Output:

```
#text =
record = suresh shinde CEO
#text =
record = ramesh deshpande Marketing Manager
#text =
record = sachin joshi Production Manager
#text =
```

6.11 THE DOCUMENT OBJECT MODEL

- The XML Document Object Model (DOM) class is a memory representation of XML Document. It defines a standard for accessing and manipulating documents.
- It allows you to programmatically read, manipulate, and modify XML documents.
- All XML elements can be accessed through XML DOM.
- It is a standard object model for XML, also a standard programming interface for XML.
- It is a World Wide Web Consortium (W3C) standard.
- According to the DOM everything in an XML document is a node. It means entire document is document node.
- Every XML element is an element node. The Text in the XML elements is text nodes, Attributes are attribute nodes, and Comments are comment node.
- In simple words, we can say XML DOM is a standard for how to get, change, add and delete XML elements.
- Under the DOM, the document is manipulated as tree broken down in nodes. It means that whole document is loaded or it is built in memory, where tree is broken down into smaller units all derived from node. Nodes are primary data type, and all other node types are derived from node.
- **Look at the following XML code of Book.xml file; we will represent hierarchical structure of it.**

Book.xml

```
<?xml version="1.0"?>
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

- Following tree diagram illustrates how memory is structured when XML data is read into DOM structure.

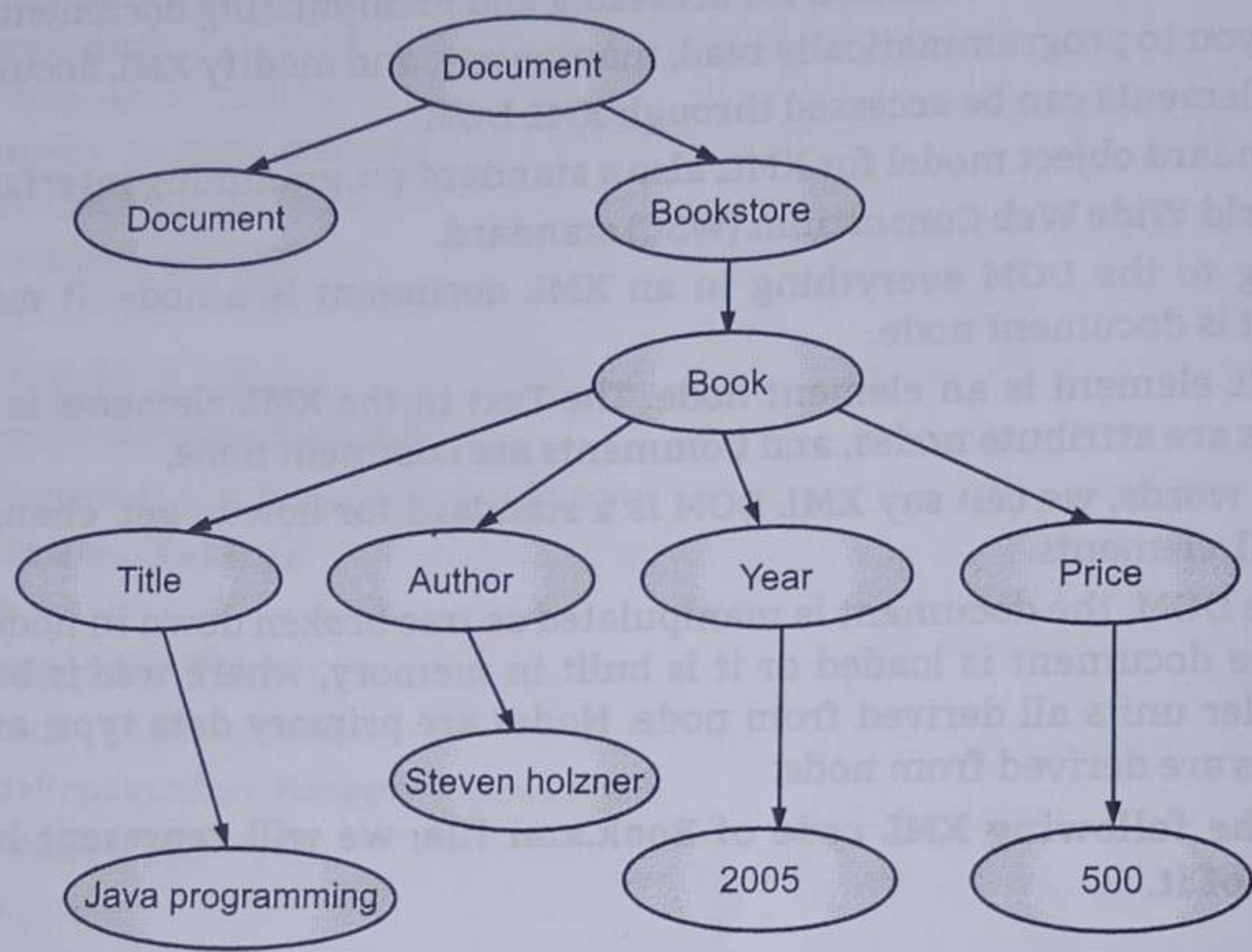


Fig. 3.1: Book.xml file tree structure

XML DOM Structure:

- Within XML document structure, each circle represents node, which is called an `XmlNode` object, The `XmlNode` object is the basic object in the DOM tree.
- The `XmlDocument` class, which extends `XmlNode`, supports methods for performing operations on document as whole.
- In addition, `XmlDocument` provides a means to view and manipulate the nodes in the entire XML `document`. Both`XmlNode` and `XmlDocument` have performance and usability enhancements and methods and properties to:
  - Access and modify nodes specific to the DOM, such as element node, entity reference node, and so on.
  - Retrieve entity nodes, in addition to the information the node contains, such as text in element node.
- Node objects have a set of methods and properties, as well as basic and well defined characteristics. Some of these characteristics are:
  - Nodes have a single parent node, a parent node being a node directly above them. The only nodes that do not have a parent is the Document root, as it is the top-level node and contains the document itself and document fragments.
  - Most nodes can have multiple child nodes, which are nodes directly below them.

- The following is a list of node types that can have child nodes:
  - **Document Node**: The node at the every top of the tree, Document is special, it is not an element, it does not correspond to anything in XML, but it represents entire document.
  - **DocumentFragment Node:** A Document Fragment is a node used for holding a part of a document, such as buffer for copy and paste. It does not need to meet all of the well formedness – for example, it could contain multiple top level elements.
  - **EntityReference:** An entity reference is an alternative name for a series of characters. You can use an entity in the &name; format, where name is the name of the entity. There are some predefined entities in XML, furthermore you can declare entities in a DTD (Document Type Definition).
  - **Element Node:** An element node in a DOM tree represents a single XML element and its contents.
  - **Attribute:** It represent a single attribute with its name, value and possibly type information. They are normally only found hidden inside element nodes, and you have to use a method such as getAttributeNode(name) to retrieve them.
  - **Text Node:** A text node is the textual content of an element.
- **Following are the XML DOM properties:**

  `x.nodeName:` It is used to get the name of x.

  `x.nodeValue:` It is used to get the value of x.

  `x.parentNode:` It is used to get the parent of node of x.

  `x.childNode:` It is used to get the child of node x.

  `x.attributes:` It is used to get the attribute node of x.
- **XML DOM Methods:**

  `x.getElementsByTagName(name):` It is used to get all the elements with a specific tag name.

  `x.appendChild(node):` It is used to insert a child node to x.

  `x.removeChild(node):` It is used to remove a child node from x, where x is a node object.

6.11.1 Creating an XML Document using the DOM

- For complex operations with PHP, we use DOM extension. This extension is enables by PHP5, it provides a sophisticated toolkit that complies with DOM Level 3 standard and provides comprehensive parsing capabilities to PHP.
- To create a XML using `DOMDocument`, basically, we need to create all the tags and attributes using the `createElement()` and `createAttribute()` methods and then create the XML structure with the `appendChild()`.
- In PHP, at first let's create an instance of `DomDocument` and initialize it, and set its version and character encoding.

```
$dom=new DomDocument('1.0','utf-8');
```

- To create a node following method is used,

```
$dom->createElement('books');
```

- To set a node as child node of another node we use following statement.

```
$dom->appendChild('previously created node');
```

Program 6.10: PHP program to create XML document using DOM.

```
<?php
$dom = new DOMDocument('1.0', 'utf-8');
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

//create the main tags, without values
$books = $dom->createElement('books');
$book_1 = $dom->createElement('book');

// create some tags with values
$name_1 = $dom->createElement('name', 'Advance PHP');
$price_1 = $dom->createElement('price', 'Rs550');
$id_1 = $dom->createElement('id', '101');

//create and append an attribute
$attr_1 = $dom->createAttribute('version');
$attr_1->value = '1.0';
//append the attribute
$id_1->appendChild($attr_1);

//$id->removeChild($att_1); it will remove child

//create the XML structure
$books->appendChild($book_1);
$book_1->appendChild($name_1);
$book_1->appendChild($price_1);
$book_1->appendChild($id_1);
$dom->appendChild($books);

//saveXML() method returns the XML in a String
print_r ($dom->saveXML());
?>
```

Output:

Advance PHP Rs550 101

Program 6.11: PHP program to demonstrate `getElementsByTagName()` method.

```
<?php
$xml = <<< XML
<?xml version="1.0" encoding="utf-8"?>
```

```
<books>
<book>Programming in Java</book>
<book>Advance PHP</book>
<book>Software Engineering</book>
</books>
XML;
$dom = new DOMDocument;
$dom->loadXML($xml);
$books = $dom->getElementsByTagName('book');
foreach ($books as $book) {
echo $book->nodeValue, PHP_EOL;
}
?>
```

Output:

Programming in Java Advance PHP Software Engineering.

6.11.2 Working with Elements

- The DOM parser works by reading an XML document and creating objects to represent the different parts of document. Each of these objects comes with specific methods and properties, which can be used to manipulate and access information about it.
- So entire document is represented as tree of these objects, with DOM parser providing a simple API to move between the different branches of the tree.

Program 6.12: PHP code to access elements of XML file using DOM.

address2.xml

```
<?xml version='1.0'?>
<address>
<street>Flat No A 201 Karve Road</street>
<county>India</county>
<city>
<name>Pune</name>
<zip>411001</zip>
</city>
<country>India</country>
</address>
```

address2.php

```
<?PHP
// initialize new DOMDocument
```

```
$doc = new DOMDocument();
// disable whitespace-only text nodes
$doc->preserveWhiteSpace = false;
// read XML file
$doc->load('address2.xml');
// get root element
$root = $doc->firstChild;
// get text node 'India
echo "Country: " . $root->childNodes->item(3)->nodeValue . "\n";
// get text node 'Pune'
echo "City: " . $root->childNodes->item(2)->childNodes->
item(0)->nodeValue . "\n";
// get text node 'Postal Code'
echo "Postal code: " . $root->childNodes->item(2)->childNodes->
item(1)->nodeValue . "\n";
?>
```

Output:

Country: India City: Pune Postal code: 411001.

- With PHP's DOM extension, we begin with initializing an instance of DOM Document object, which represents an XML document. Once object is initialized, is used to parse XML file through load() function, which accepts the path of XML file.
- The result of the load() method is a tree containing `DOMNode` objects, with every object exposing various properties and methods for accessing its parent, child, and sibling nodes.
- For example, every `DOMNode` object exposes a `parentNode` property, which can be used to access its parent node, and a `childNodes` property, which returns a collection of its child nodes. Every `DOMNode` object also exposes `nodeName` and `nodeValue` properties, which can be used to access the node's name and value respectively. It's thus quite easy to navigate from node to node of the tree, retrieving node values at each stage.
- To illustrate the process, consider the preceding script carefully. Once the XML document has been loaded, it calls the `DOMDocument` object's `firstChild` property, which returns a `DOMNode` object representing the root element <address>.
- This `DOMNode` object, in turn, has a `childNodes` property, which returns a collection of all the child elements of <address>. Individual elements of this collection can be accessed via their index position using the item() method, with indexing starting from zero. These elements are again represented as `DOMNode` objects; as such, their names and values are therefore accessible via their `nodeName` and `nodeValue` properties.
- The element `<name>`, which is the first child of the `<city>` element, is accessible via the path `$root->childNodes->item(2)->childNodes->item(0)`, and the text value 'Oxford' is accessible via the path `$root->childNodes->item(2)->childNodes->item(0)->nodeValue`.

6.12 THE SIMPLE XML EXTENSION

- PHP supports for both Document Object Model and SAX parsing methods. The easiest way to work with XML data in PHP is through its Simple XML Extension. It was enabled by default in PHP5, it provides a user friendly and intuitive interface to reading and processing XML documents.
- Simple XML represents every XML document as an object and turns the element within it into hierarchical set of objects and object properties. Accessing elements becomes simple as parent→child notation to traverse the object tree until that element is reached.
- `SimpleXML` is an extension that allows us to easily manipulate and get XML data. `SimpleXML` provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.
- `SimpleXML` turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

`SimpleXMLElement` methods that you can use to manipulate XML documents:

3. `addAttribute(name, value)`: Adds an attribute named name, with the value, to the element.
3. `addChild(name [, value])`: Adds a child element called name to the element. The child element can be empty, or it can contain the text value. It returns the child element as a new `SimpleXMLElement` object.
3. `asXML([filename])`: Generates an XML document from the `SimpleXMLElement` object. If filename is supplied, it writes the XML to the file; otherwise it returns the XML as a string.
3. `attributes()`: Returns an associative array of all the attributes in the element, as name=>value pairs.
3. `children()`: Returns an array of all the element's children, as `SimpleXMLElement` objects.
3. `getName()`: Returns the name of the element as a string.
3. `xpath(path)`: Finds child elements that match the given XPath (XML Path Language) path string.

`SimpleXML` gives you three functions that you can use to import XML data into a `SimpleXMLElement` object:

0. `simplexml_import_dom(node)`: Converts the supplied DOM node, into a `SimpleXMLElement` object.
0. `simplexml_load_file(filename)`: Loads the XML file with name filename as a `SimpleXMLElement` object.
0. `simplexml_load_string(string)`: Loads the supplied XML string as a `SimpleXMLElement` object.

6.12.1 Reading an XML Elements

Reading XML elements using `simpleXML`:

- You can use `simplexml_load_file` function to load external XML file in PHP program and create an object, then you can access any element from XML by using this object as shown in below.

Program 6.13: Program to illustrate simple `XML_extension`.

Address.xml

```
<?xml version='1.0'?>
<address>
<Flat_No>A 202</Flat_No>
<Street>KARVE ROAD</Street>
<region>
<name>PUNE</name>
<zip>411020</zip>
<state>Maharashtra</state>
<country>INDIA</country>
</region>
</address>
```

address.php

```
<?php
// to load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/Address.xml') or die
   ("Unable to load XML!");
echo "City Name: " . $xml->region->name . "<br>";
echo "ZIP Code: " . $xml->region->zip . "<br>";
?>
```

Output:

City Name: PUNE

ZIP Code: 411020

- In the above example, we have read XML file `address.xml` with `simple_xml_load_file()` function. This function reads and parse XML file and assuming it is well formed return simple XML object representing document root element. This object is top level of hierarchical object tree that mirrors internal structure of XML data. The elements below the root are represented with parent-> child relationship.
- While working with attributes, Simple XML has easiest way to get access to attributes. These are converted into keys and values of a PHP associative array and they can be accessed like an array elements.

6.12.2 Reading XML Elements as Loop

- We can use foreach loop to iterate through the entire XML file and read elements from XML. For each loop access all the children of an object.
- We can access the XML file contents using `simplexml_load_file()` nfunction and assign it to $xml object and then we can pass this $xml object to foreach loop and read all the elements of the file using loop.

Program 6.14: PHP program to demonstrate `simplexml_load_file()` and display contents using for each method.

Book.xml

```
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

book.php

```
<?php
// load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/Book.xml') or die
  ("Unable to load XML!");
foreach ($xml->book as $b) {
echo "Book Title=".$b->title . "<br>";
echo "Book Author=".$b->author . "<br>";
echo "Book year=".$b->year . "<br>";
echo "Book price=".$b->price . "<br>";
}
?>
```

Output:

```
Book Title=Java programming
Book Author=Steven holzner
Book year=2005
Book price=500.00
Book Title=Harry Potter
Book Author=J K. Rowling
Book year=2005
Book price=450
Book Title=Mine kamph
Book Author=Adolf hitler
Book year=1950
Book price=525.00
```

- In above program, for each loop iterates over book elements in XML data turning each into object, attributes of book element are represented as elements of associative array.

6.12.3 Creating an XML Document with SimpleXML

Creating New XML Documents:

- We can also use `SimpleXML` to create new XML elements from scratch, by initializing an empty `SimpleXML` object from an XML String, and then using `addchild()` and `addAttribute()` methods to build the rest of the XML document tree.

Program 6.15: PHP code for creating XML document with SimpleXML.

```
<?php
// load XML from string
$xmlStr = "<?xml version='1.0'?><person></person>";
$xml = simplexml_load_string($xmlStr);
// add attributes
$xml->addAttribute('age', '25');
$xml->addAttribute('gender', 'male');
// add child elements
$xml->addChild('name', 'santoshvarma');
$xml->addChild('dob', '01-01-2000');
// add second level of child elements
$address = $xml->addChild('address');
$address->addChild('street', '12 East Bandra Road');
$address->addChild('city', 'Mumbai');
// add third level of child elements
```

```
$country = $address->addChild('country', 'India');
$country->addAttribute('code', '+91');
// output new XML string
header('Content-Type: text/xml');
echo $xml->asXML();
?>
```

Output:

```
-<person age="25" gender="male">
  <name>santosh varma</name>
  <dob>01-01-2000</dob>
 -<address>
    <street>12 East Bandra Road</street>
    <city>Mumbai</city>
    <country code="+91">India</country>
  </address>
</person>
```

- This script begins by initializing a string variable to hold the XML document prolog and root element. The `simple_xml_load_string()` method take care of converting string into `SimpleXML` object representing the document's root element.
- Once this object has been initialized, it's simple to add child elements and attributes to it, and to build the rest of the XML document tree programmatically.

6.13 CHANGING A VALUE WITH SIMPLE XML

- With the Simple XML we can easily change the content in an XML file, we have to simply assign a new value to the corresponding object property using = operator.

Program 6.16: PHP program for changing the values in XML.

contactinfo.xml

```
<?xml version="1.0"?>
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
```

```
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

simplexmlexample.PHP

```
<?php
// load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/contactinfo.xml') or
   die ("Unable to load XML!");
$xml->book[1]->title = 'C++ programming';
$xml->book[1]->author = 'E Balguruswamy';
// output new XML string
header('Content-Type: text/xml');
echo $xml->asXML();
?>
```

Output:

```
-<bookstore>
 -<book category="programming">
    <title lang="en">Java programming</title>
    <author>Steven holzner</author>
    <year>2005</year>
    <price>500.00</price>
  </book>
 -<book category="CHILDREN">
    <title lang="en">C++ programming</title>
    <author>E Balguruswamy</author>
    <year>2005</year>
    <price>450</price>
  </book>
 -<book category="literature">
    <title lang="en">Mine kamph</title>
    <author>Adolf hitler</author>
    <year>1950</year>
    <price>525.00</price>
  </book>
</bookstore>
```

- As you can see in output, we have assigned new values to the book elements attributes title and author having index value 1. Here we have used as `XML()` method which converts the nested hierarchy of `simpleXML` objects and object properties back into a regular XML string.

6.13.1 Adding new Attributes and Child's to the XML document

- We can add attributes to different elements of XML by using `addAttributes()` function. Also we can add new childs to elements by using `addchild()` method.

Program 6.17: Php program for adding new attributes and child's.

contactinfo.xml

```
<?xml version="1.0"?>
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

contactinfo.php

```
<?php
// load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/contactinfo.xml') or
   die ("Unable to load XML!");
$book = $xml->addChild('book');
// add 'region' attribute
$book->addAttribute('region','India');
// add <title>, <author> and <year> elements
$title = $book->addChild('title', 'C Programming');
$author = $book->addChild('author', 'YashwantKanetkar');
$page = $book->addChild('year', '2000');
// output new XML string
header('Content-Type: text/xml');
echo $xml->asXML();
?>
```

Output:

```
▼<bookstore>
 ▼<book category="programming">
    <title lang="en">Java programming</title>
    <author>Steven holzner</author>
    <year>2005</year>
    <price>500.00</price>
  </book>
 ▼<book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>450</price>
  </book>
 ▼<book category="literature">
    <title lang="en">Mine kamph</title>
    <author>Adolf hitler</author>
    <year>1950</year>
    <price>525.00</price>
  </book>
 ▼<book region="India">
    <title>C Programming</title>
    <author>YashwantKanetkar</author>
    <year>2000</year>
  </book>
</bookstore>
```

- Simple XML object exposes an `addChild()` method for adding new child elements and `addAttribute()` method for adding new attributes. Both of these methods accept name and value, generates the corresponding element or attribute, and attach it to parent object with hierarchy.
- As you can see in the above output, new attribute region is added and we have set new values to title author and year.

PROGRAMS

Program 6.18: Write AJAX program to get player details from XML file when user select a player name. Create XML file for storing details of player (Country, player name, wickets, runs).

player.xml:

```
<?xml version="1.0"?>
<playerinfo>
    <player>
    <Country>India</Country>
    <Player_name>Sachin</Player_name>
```

```
<Wickets>9</Wickets>
<Runs>450</Runs>
</player>
<player>
<Country>India</Country>
<Player_name>Dhoni</Player_name>
<Wickets>19</Wickets>
<Runs>200</Runs>
</player>
<player>
<Country>India</Country>
<Player_name>Virat</Player_name>
<Wickets>6</Wickets>
<Runs>321</Runs>
</player>
</playerinfo>
```

xmlplayer.html:

```
<html>
<head>
<script>
function show(str)
{
   if(window.XMLHttpRequest)
   {
      ob=new XMLHttpRequest();
   }
   else
   {
      ob=new ActiveXObject("Microsoft.XMLHTTP");
   }
   ob.onreadystatechange=function()
   {
      if(ob.readyState==4 && ob.status==200)
      {
         document.getElementById("a").innerHTML=ob.responseText;
      }
   }
   ob.open("GET","AJAX xmlplayer.php?q="+str,true);
   ob.send();
```

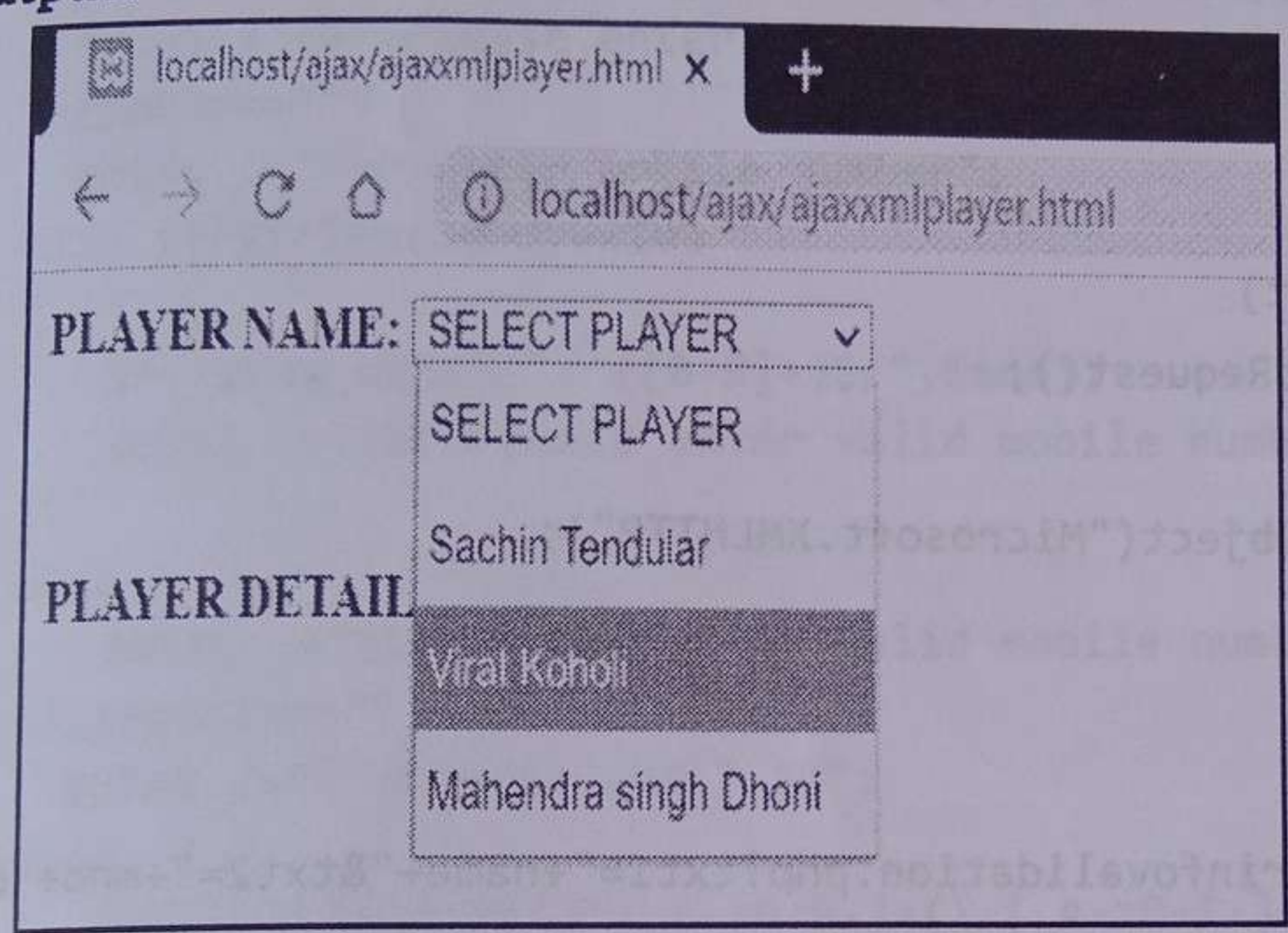
```
}
</script>
</head>
<body>
<form>
<b>PLAYER NAME:</b>
<select name="player_name" onchange="show(this.value)">
<option value="">SELECT PLAYER </option>
<option value="Sachin">Sachin Tendular</option>
<option value="Virat">Virat Koholi</option>
<option value="Dhoni">Mahendra singh Dhoni</option>
</select>
</form>
<br><br>
<div id="a"><b>PLAYER DETAILS : </b></div>
</body>
</html>
```

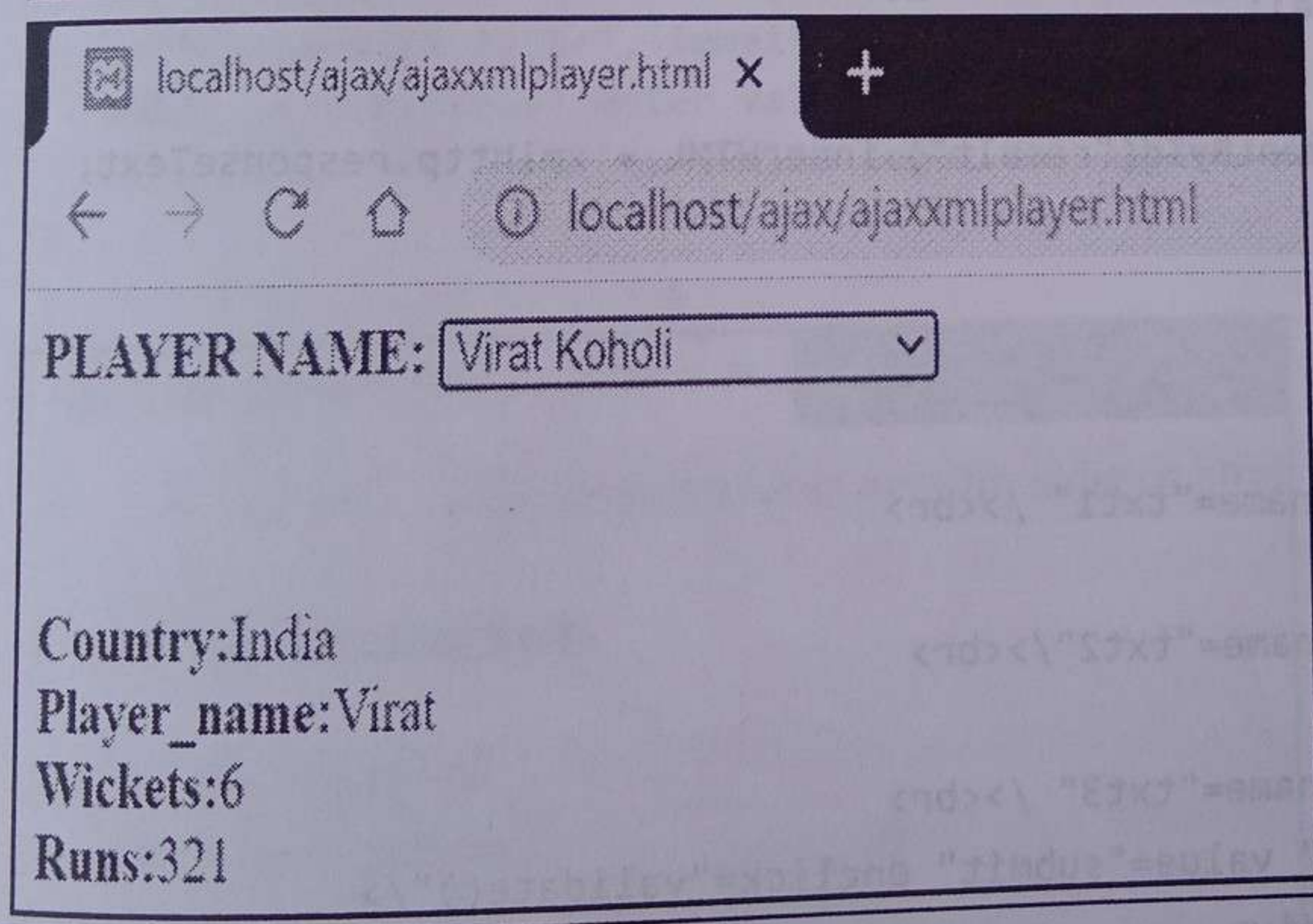
xmlplayer.php:

```
<?php
$q=$_GET["q"];
$dom=new DOMDocument();
$dom->load("player.xml");
$x=$dom->getElementsByTagName('Player_name');
for($i=0;$i<=$x->length-1;$i++)
{
    if($x->item($i)->nodeType==1)
    {
        if($x->item($i)->childNodes->item(0)->nodeValue==$q)
        {
            $y=($x->item($i)->parentNode);
        }
    }
}
$player=($y->childNodes);
for($i=0;$i<$player->length;$i++)
{
    if($player->item($i)->nodeType==1)
    {
```

```
        echo "<b>".$player->item($i)->nodeName.":</b>";
        echo $player->item($i)->childNodes->item(0)->nodeValue;
        echo "<br>";
    }
}
?>
```

Output:





- The user will get dropdown list of player names. When user selects any name from the list, then using PHP script name of particular player will be checked into `player.xml` file and details will be displayed.

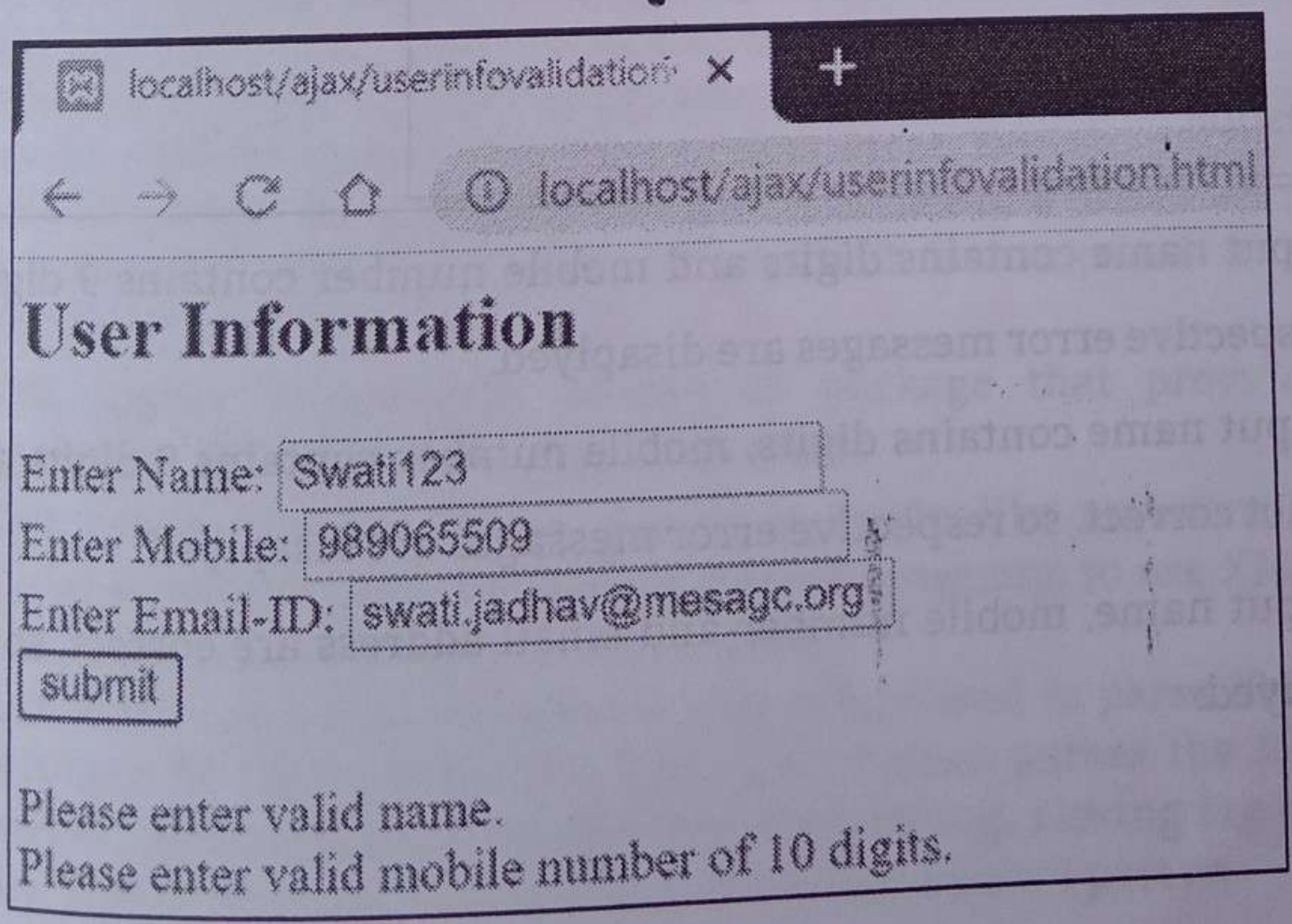
Program 6.19: Write a PHP script using AJAX concept, to check name, mobile number and email address are valid or invalid.

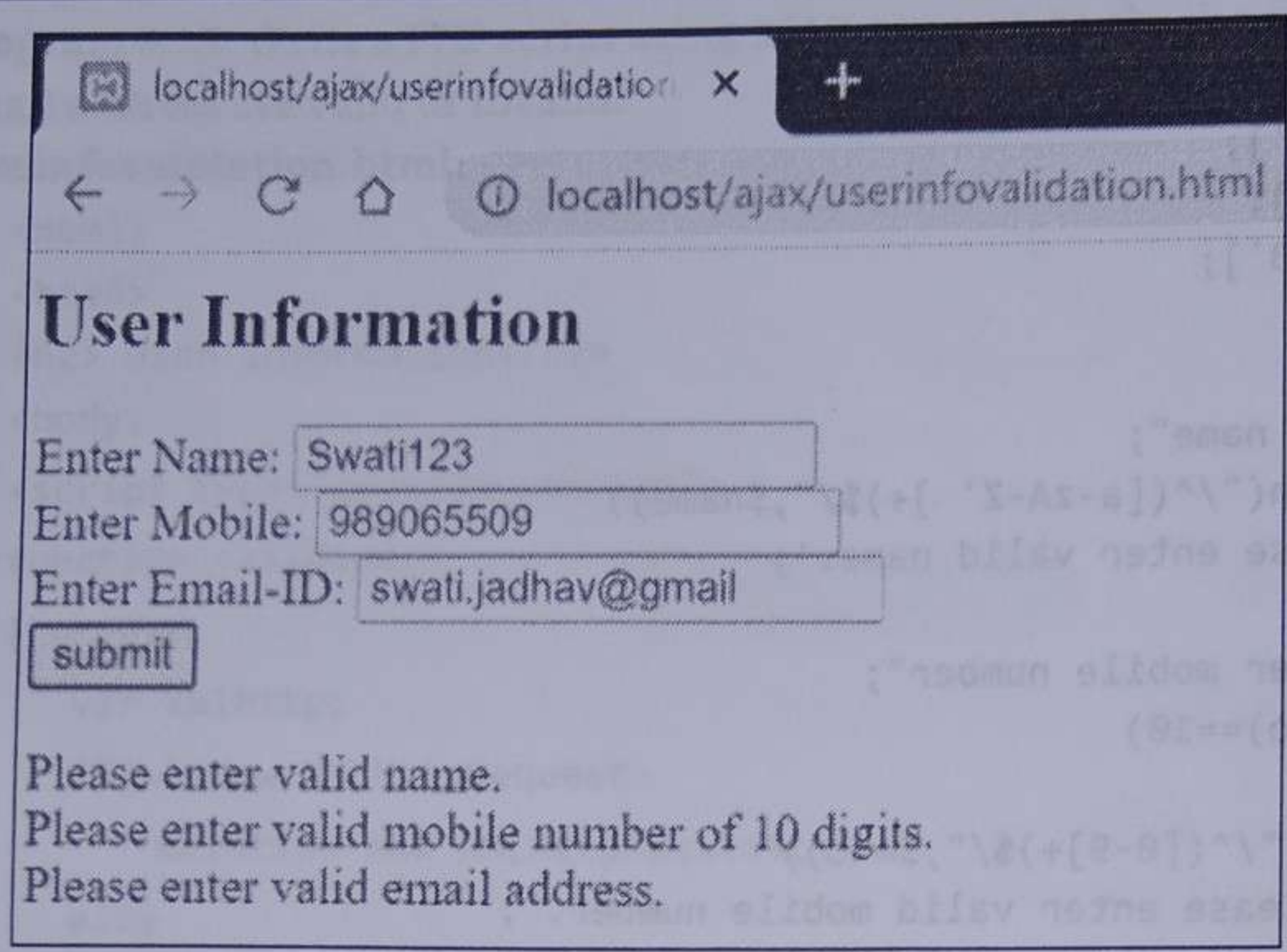
userinfovalidation.html:

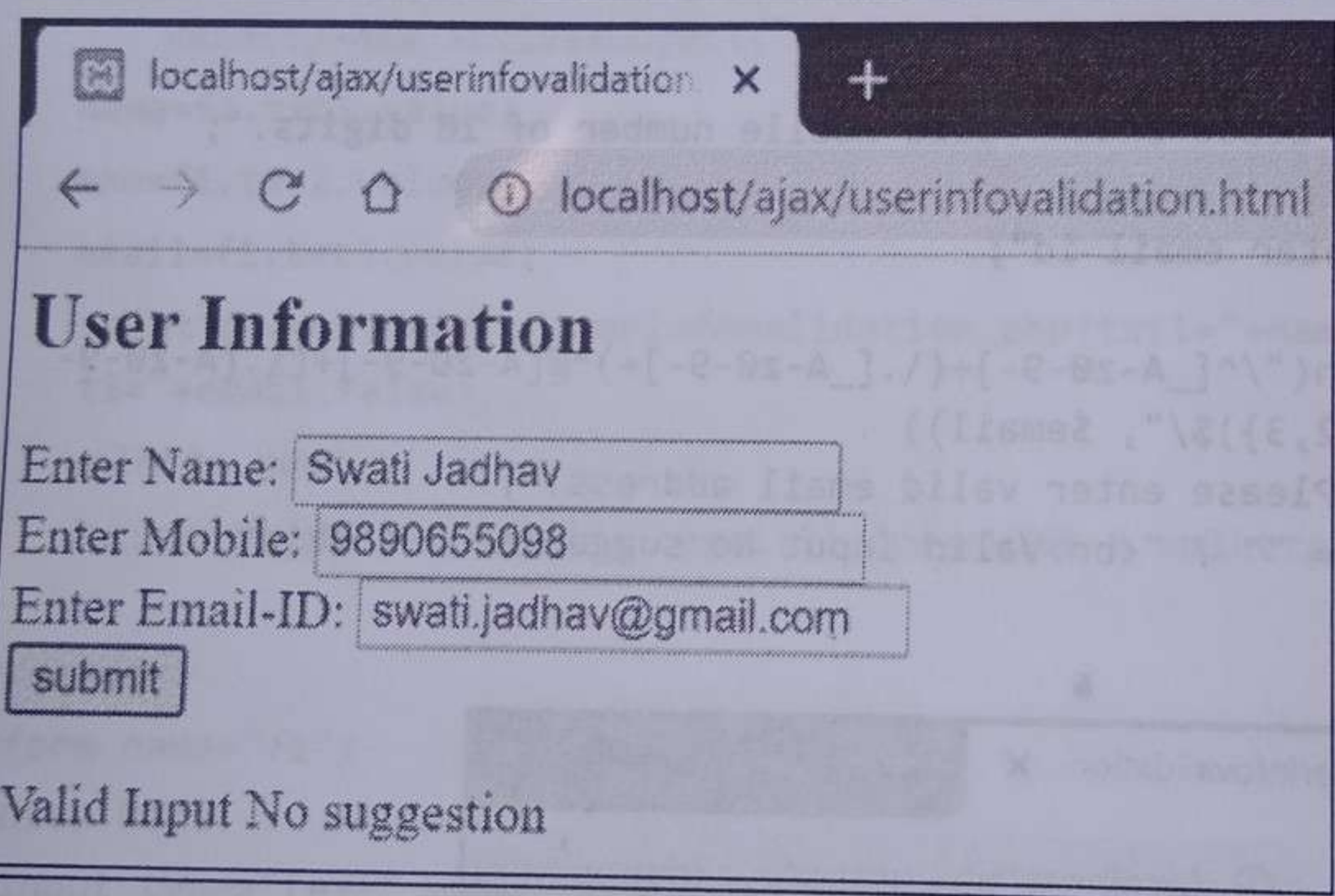
```
<html>
<head>
<h2> User Information</h2>
<body>
<script type="text/javascript">
function validate()
{
    var xmlhttp;
    if(window.XMLHttpRequest)
        xmlhttp= new XMLHttpRequest();
    else
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    name=f1.txt1.value;
    mno=f1.txt2.value;
    email=f1.txt3.value;
    xmlhttp.open("GET","userinfovalidation.php?txt1="+name+"&txt2="+mno+"&tx
    t3="+email,false);
    xmlhttp.send();
    document.getElementById("result").innerHTML = xmlhttp.responseText;
}
</script>
<form name="f1">
Enter Name:
<input type="text" name="txt1" /><br>
Enter Mobile:
<input type="text" name="txt2"/><br>
Enter Email-ID:
<input type="text" name="txt3" /><br>
<input type="button" value="submit" onclick="validate()"/>
<div id="result"></div>
</form>
</body>
</html>
```

userinfovalidation.php:

```
<?php
$name= $_GET['txt1'];
$mno= $_GET['txt2'];
$email= $_GET['txt3'];
$hint="";
if($name=="")
  $hint="<br>Enter name";
else if(!preg_match("/^([a-zA-Z' ]+)$/",$name))
  $hint ='<br>Please enter valid name.';
if($mno=="")
  $hint .="<br>Enter mobile number";
else if(strlen($mno)==10)
{
   if(!preg_match("/^([0-9]+)$/",$mno))
   $hint .='<br>Please enter valid mobile number.';
}
else
   $hint .='<br>Please enter valid mobile number of 10 digits.';
if($email=="")
  $hint .="<br>Enter email id";
else
   if(!preg_match("/^[_A-z0-9-]+(\.[_A-z0-9-]+)*@[A-z0-9-]+(\.[A-z0-9-
   ]+)*(\.[A-z]{2,3})$/", $email))
   $hint .='<br>Please enter valid email address.';
   echo $hint === "" ? "<br>Valid Input No suggestion" : $hint;
?>
```







- **1st Output Screen:** Input name contains digits and mobile number contains 9 digits, which is invalid. so respective error messages are disaplyed.
- **2nd Output Screen:** Input name contains digits, mobile number contains 9 digits and email address is also not correct. so respective error messages are disaplyed.
- **3rd Output Screen:** Input name, mobile number and email address are correct, so no error message is displayed.

Summary

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating dynamic, faster and interactive web-based applications using XML, HTML, CSS, and Java Script.
- AJAX uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript.
- The request to the serve can be synchronous or asynchronous. Synchronous request blocks the user until a response is retrieved whereas asynchronous doesn't block the user.
- `XMLHttpRequest` object's `open()` and `send()` methods are used to send request to server.
- The browser can request database information to be retrieved, XML file data, text file, form data validation etc.
- AJAX can be used for interactive communication with XML and database.
- AJAX has two applications models i.e. synchronous and asynchronous.
- `XMLHttpRequest` is used for asynchronous communication between client and server.
- The `XMLHttpRequest` object can send HTTP request, and receive responses.
- There are various PHP frameworks available, like AJAX Core, CakePHP, XAJAX, SAJAX, XOAD, Zephyr, Feather AJAX 1.1 and Tigermouse to integrate AJAX with PHP.
- The SAJAX is an AJAX based framework which generates AJAX enabled JavaScript from many server side languages like PHP, ASP, ColdFusion, Io, Lua, Perl, Python and Ruby.
- XML it is a markup language used for storing and transporting `data.XML` doesn't depend upon platform and software. XML stands for Extensible Markup Language. You must write valid XML tags which have opening and closing tags. You can use PHP with XML with several methods which are used to generate, read XML data and used to Display XML data.
- XML Document has tree structure, where root element is at top and all the child elements are connected to root element.
- Document Object Model is programming interface for HTML and XML documents. It is used to define logical structure of document. Documents are modeled using objects, and the model includes not only the structure of a document but also the behavior of a document and the objects of which it is composed of like tag elements with attributes in HTML.
- XML parser is software library or package that provides interface for client applications to work with XML document.
- XML parser is event based parser, it works like an event handler. XML parser is designed to read XML and create way of programs to use XML. It validates document and check the document is well formed.
- SAX Parser and DOM Parser are commonly used to parse XML Documents. SAX is an acronym for Simple API for XML. SAX Parser parses the XML file line by line and triggers events when it encounters opening tag, closing tag or character data in XML file. This is why SAX parser is called an event-based parser.

- DOM is an acronym for Document Object Model. Unlike SAX parser DOM parser loads the complete XML file into memory and creates a tree structure where each node in the tree represents a component of XML file. With DOM parser you can create nodes, remove nodes, change their contents and traverse the node hierarchy.
- `SimpleXML` is an extension that allows us to easily manipulate and get XML data.
- `SimpleXML` provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.
- `SimpleXML` turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

Check Your Understanding

Q. I Multiple choice questions.

1. How can you find out that an AJAX request has been completed?
   (a) readyState=4 (b) readyState=3
   (c) readyState=1 (d) readyState=2
2. What is the name of object used for AJAX request?
   (a) XmlHttpRequest (b) XMLHttpRequest
   (c) XMLHTTPRequest (d) xmlhttprequest
3. AJAX Stands for_____.
   (a) Abstract JSON and XML
   (b) Another Java Abstraction for X-Windows
   (c) Another Java and XML Library
   (d) Asynchronous Javascript and XML
4. The responseText property returns the response as a string format
   (a) True (b) False
5. XML stands for _____.
   (a) Extension modern link (b) extensible markup language
   (c) extra modern language (d) x-markup language
6. XML comments are written as _____.
   (a) <! – !> (b) <!--- ?
   (c) <? – ?> (d) <$----%>
7. Which of the following is valid XML tag?
   (a) <? Xml version="1.0"?> (b) <$ xml version=1.2">
   (c) <xml version="1.0"> (d) None of above
8. DTD means?
   (a) Data Type Definition. (b) Data Text Decode.
   (c) Define Text Data. (d) None of above.
9. XML DOM object is____.
   (a) Entity. (b) Entity Reference.
   (c) Comment Reference. (d) Comment Data.
10. Which of the following is XML Parser?
   (a) SAX parser (b) DOM Parser
   (c) CDATA Parser (d) (a) and (b)

ANSWERS

1. (a)	2. (b)	3. (d)	4. (a)	5. (b)	6. (b)	7. (a)	8. (a)
9. (b)	10. (d)						

Q II. States True / False.

(1) We can frame our XML tags.

(2) XML is having World Wide Web Consortium (W3C) recommendation.

(4) XML tags must have opening and closing tags.

(5) XML stands for Extensible Meta Language.

(6) AJAX is a program.

(7) HTTP the server support of AJAX.

(8) Asynchronous Java Script and XML, is full form of AJAX.

(9) AJAX can work with web application.

(10) XMLHTTPRequest is not used by AJAX.

(11) XML is used to replace HTML.

(12) DTD stands for Document Data Definition.

ANSWERS

1. True	2. True	3. True	4. True	5. True	6. True
7. True	8. True	9. True	10. True	11. False	12. False

Practice Questions

Q. I Answer the following questions in short.

1. Which are different technologies used in AJAX ?
2. Where AJAX cannot be used?
3. What are the advantages of AJAX ?
4. How can you find out that an AJAX request has been completed?
5. What is the difference between synchronous and asynchronous requests?
6. What are the security issues with AJAX ?
7. Write syntax for creating XMLHttpRequest object.
8. What are different possible value for readyState property?
9. Which methods are used in AJAX to send the request to server?
10. List types of XML parser.
11. List any two methods used wit SimpleXML.
12. What is meaning of valid and invalid XML tags?

13. List XML elements.
14. What are different node types in DOM?
15. What is DOMDocument()?

Q. II Answer the following questions.

1. What are the differences between AJAX and JavaScript?
2. How GET and POST methods are used in AJAX ? Explain in detail with syntax.
3. Explain properties of XMLHttpRequest object.
4. Write AJAX program to accept username and password from user. Also validate username and password from database using PHP script. If both are matching in the database then display "Login Successful" otherwise display "Invalid Login details".
5. Write AJAX program to accept book title from user. When user clicks submit button, display book details from XML file if matches otherwise display "No Match Found" using PHP.
6. Write AJAX program to request an XML file from server and print the details of books when user clicks on the Print button.
7. Write AJAX program to fetch suggestions when is user is typing in a textbox. (Hint create array of suggestions and matching string will be displayed).
8. Explain Concept of XML?
9. What is Document Object Model in PHP?
10. What is XML parser? What are different types of it?
11. What is SimpleXML extension?
12. Write a PHP code to display XML data through SimpleXML.
13. Write a PHP code to generate XML.
14. Write PHP script to create a CD catalog using XML file.
15. Create a XML file which gives details of books available in "ABC Bookstore" from following categories

    (1) Technical, (2) Cooking, (3) YOGA
16. Create an application that reads "Book.xml" file into simple XML object. Display attributes and elements(Hint: use simple_xml_load_file() function).

Q.III Define the terms:

1. Synchronous communication
2. Asynchronous communication
3. responseXML
4. responseText
5. XMLHttpRequest Object
6. DTD.
7. XML Schema
8. element
9. Attribute

***