

**THIRD YEAR DIPLOMA  
ENGINEERING AND TECHNOLOGY**

**COMPUTER ENGINEERING AND  
INFORMATION TECHNOLOGY GROUP**

**SEMESTER-VI**



# **WEB BASED APPLICATION DEVELOPMENT WITH PHP**



**PRASHANT D. SOMWANSHI**  
**Mrs. MANISHA A. POKHARKAR**  
**Mrs. MRUNAL P. FATANGARE**

# Syllabus ...

## 1. Expressions and Control Statements in PHP

- 1.1 History and Advantages of PHP, Syntax of PHP.
- 1.2 Variables, Data Types, Expressions and Operators, Constants.
- 1.3 Decision Making Control Statements - if, if-else, Nested if, switch, break and continue Statement.
- 1.4 Loop Control Structures - while, do-while, for and foreach.

## 2. Arrays, Functions and Graphics

- 2.1 Creating and Manipulating Array, Types of Arrays - Indexed, Associative and Multi-Dimensional Arrays.
- 2.2 Extracting Data from Arrays, Implode, Explode, and Array Flip.
- 2.3 Traversing Arrays
- 2.4 Function and its Types - User Defined Function, Variable Function and Anonymous Function.
- 2.5 Operations on String and String Function : str\_word\_count(), strlen(), strtolower(), strtoupper(), str\_replace(), ucwords(), strtolower(), strtolower(), strcmp().
- 2.6 Basic Graphics Concepts, Creating Images, Images with Text, Scaling Images, Creation of PDF Document.

## 3. Apply Object Oriented Concepts in PHP

- 3.1 Creating Classes and Objects.
- 3.2 Constructor and Destructor.
- 3.3 Inheritance, Overloading and Overriding, Cloning Object.
- 3.4 Introspection, Serialization.

## 4. Creating and Validating Forms

- 4.1 Creating a Webpage using GUI Components, Browser Role - GET and POST Methods, Server Role.
- 4.2 Form Controls: Textbox, Textarea, Radiobutton, Checkbox, List, Buttons.
- 4.3 Working with multiple Forms:
  - A Web Page having Many Forms.
  - A Form having Multiple Submit Buttons.
- 4.4 Web Page Validation.
- 4.5 Cookies - Use of Cookies, Attributes of Cookies, Create Cookies, Modify Cookies Value, and Delete Cookies.
- 4.6 Session - Use of Session, Start Session, Get Session Variables, Destroy Session.
- 4.7 Sending E-mail.

## 5. Database Operations

- 5.1 Introduction to MySQL - Create a Database.
- 5.2 Connecting to a MySQL Database: MySQL Database Server from PHP.
- 5.3 Database Operations: Insert Data, Retrieving the Query Result.
- 5.4 Update and Delete Operations on Table Data.



# Contents ...

<b>1. Expression and Control Statements in PHP</b>		<b>1.1 - 1.30</b>
1.0	Introduction	1.1
1.1	History, Advantages and Syntax of PHP	1.3
1.1.1	History of PHP	1.3
1.1.2	Advantages and Disadvantages of PHP	1.4
1.1.3	Syntax of PHP	1.4
1.1.4	Step to Run Program using XAMPP Server	1.5
1.2	Variables	1.6
1.2.1	Data Types	1.9
1.2.2	Expressions and Operators	1.13
1.2.2.1	Expressions	1.13
1.2.2.2	Operators	1.13
1.2.3	Operator Precedence and Associativity	1.16
1.2.4	Constants	1.17
1.3	Decision making Control Statements	1.17
1.3.1	if Statement	1.17
1.3.2	if-else Statement	1.18
1.3.3	Nested if Statement	1.21
1.3.4	switch Statement	1.22
1.3.5	break Statement	1.24
1.3.6	continue Statement	1.25
1.4	Loop and Control Structures	1.26
1.4.1	while Loop	1.27
1.4.2	do while Loop	1.28
1.4.3	for Loop	1.28
1.4.4	foreach Loop	1.29
*	Practice Questions	1.30
<b>2. Arrays, Functions and Graphics</b>		<b>2.1 - 2.24</b>
2.0	Introduction	2.1
2.1	Creating and Manipulating Array	2.1
2.1.1	Types of Arrays	2.2
2.1.1.1	Indexed Array	2.2
2.1.1.2	Associative Array	2.2
2.1.1.3	Multi-Dimensional Array	2.3
2.2	Extracting Data from Arrays	2.3
2.2.1	Implode	2.4
2.2.2	Explode	2.5
2.2.3	Array Flip	2.6
2.3	Traversing Arrays	2.6
2.4	Function and its Types	2.10
2.4.1	Defining and Calling a Function	2.10
2.4.1.1	Defining a Function	2.10
2.4.1.2	Calling a Function	2.11
2.4.1.3	Function Arguments	2.11
2.4.1.4	Returning Values	2.12

2.4.1.5	Function Body	2.13
2.4.2	Function Types	2.13
2.4.2.1	User Defined Functions	2.13
2.4.2.2	Variable Function	2.13
2.4.2.3	Anonymous Function (Lambda Function)	2.14
2.5	Operations on String and String Functions	2.14
2.5.1	String Functions/String Operations	2.17
2.5.1.1	str_word_count()	2.17
2.5.1.2	strlen()	2.17
2.5.1.3	strrev()	2.18
2.5.1.4	strpos()	2.18
2.5.1.5	strrpos()	2.18
2.5.1.6	str_replace()	2.18
2.5.1.7	ucwords()	2.19
2.5.1.8	strtoupper()	2.20
2.5.1.9	strtolower()	2.20
2.5.1.10	strcmp()	2.20
2.6	Basic Graphics Concepts	2.20
2.6.1	Creating Images	2.21
2.6.2	Images with Text	2.22
2.6.3	Scaling Images	2.22
2.6.4	Creation of PDF Document	2.23
*	Practice Questions	2.24

### **3. Apply Object Oriented Concepts in PHP** **3.1 – 3.14**

3.0	Introduction	3.1
3.1	Creating Classes and Objects	3.2
3.1.1	Creating Classes	3.2
3.1.2	Creating Object	3.3
3.2	Constructor and Destructor	3.4
3.2.1	Constructor	3.4
3.2.2	Destructor	3.6
3.3	Inheritance	3.6
3.3.1	Method of Function Overloading	3.7
3.3.2	Method of Function Overriding	3.8
3.3.3	Cloning Object	3.9
3.4	Introspection	3.10
3.4.1	Examining Classes	3.10
3.4.2	Examining an Object	3.11
3.5	Serialization	3.11
*	Practice Questions	3.13

### **4. Creating and Validating Forms** **4.1 – 4.22**

4.0	Introduction	4.1
4.1	Creating a Webpage using GUI Components	4.2
4.1.1	Browser Role GET and POST Methods	4.2
4.1.2	Server Role	4.4
4.2	Form Controls	4.5
4.2.1	Textbox	4.5
4.2.2	Textarea	4.6

4.2.3	Radio Button	4.6
4.2.4	Checkbox	4.7
4.2.5	List	4.8
4.2.6	Buttons	4.9
4.3	Working with Multiple Forms	4.10
4.3.1	Web Page having Many Forms	4.10
4.3.2	Form having Multiple Submit Buttons	4.11
4.4	Web Page Validation	4.12
4.5	Cookies	4.14
4.5.1	Types of Cookies	4.15
4.5.2	Use of Cookies	4.15
4.5.3	Attribute of Cookies	4.16
4.5.4	Create Cookies	4.16
4.5.5	Modify Cookies Value	4.17
4.5.6	Delete Cookies	4.17
4.6	Session	4.18
4.6.1	Use of Session	4.18
4.6.2	Start Session	4.19
4.6.3	Get Session Variables	4.19
4.6.4	Destroy Session	4.20
4.7	Sending E-mail	4.20
*	Practice Questions	4.22

## 5. Database Operations

5.1 – 5.8

5.0	Introduction	5.1
5.1	Introduction to MySQL	5.1
5.1.1	Create a Database	5.3
5.2	Connecting to a MySQL Database	5.4
5.2.1	MySQL Database Server from PHP	5.4
5.2.1.1	mysqli_connect() Function	5.4
5.2.1.2	PDO::__construct() Function	5.4
5.3	Database Operations	5.5
5.3.1	Insert Data	5.5
5.3.2	Retrieving the Query Result	5.6
5.4	Update and Delete Operations on Table Data	5.7
5.4.1	Update Data	5.7
5.4.2	Delete Data	5.7
*	Practice Questions	5.8

## Programs

P.1 – P.26



# 1...

## Expressions and Control Statements in PHP

### Chapter Outcomes...

- Write simple PHP program to solve the given expression.
- Use relevant decision making control statement to solve the given problem.
- Solve the given iterative problem using relevant loop statement.

### Learning Objectives...

- To learn Web Based Application Development using PHP
- To understand Basic Concepts of PHP
- To study History, Syntax, Data Types, Variables etc., in PHP
- To learn Decision Making Control Statements in PHP
- To study Loop Control Structures in PHP

### 1.0 INTRODUCTION

- A Web application is a computer software or program that performs some specific tasks at its client by using a Web browser. The Web-based applications are also known as Web apps.
- Web applications are usually based on the client-server architecture where the client input/request data while the server stores and responds with result.
- Fig. 1.1 shows concept of Web application. User/client triggers a request to the web server over the Internet, either through a web browser or the Application's User Interface (API). Web server forwards this client/user request to the appropriate web application server.
- The Web application server performs the requested task - such as querying the database or processing the data - then generates the results of the requested data. Web application server sends results to the web server with the requested information or processed data.
- Web server responds back to the client/user with the requested information that then appears on the user's display. In short, a Web application is a program that runs on a Web server while user/client accesses it using Web browser.

#### Advantages of using Web-Based Applications:

1. Web-based apps are cross-platform and universally accessible.
2. Web-based applications are highly scalable.
3. Web-based apps deployment is easy, cost-effective, and fast.
4. Web-based applications are easy to update and maintain.

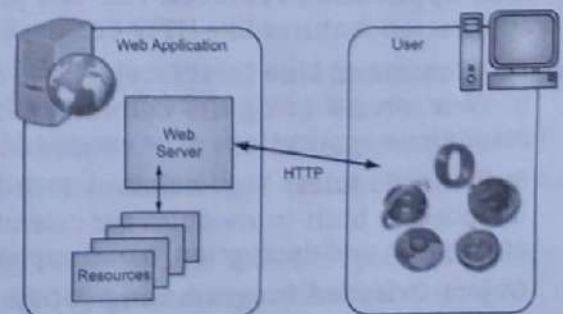


Fig. 1.1: Concept of Web Application (App)

[1.1]

- Web applications use a combination of server-side scripts like PHP to handle the storage and retrieval of the information and client-side scripts like JavaScript and/or HTML to present information to users.
- A script is a set of programming instructions that is interpreted at runtime. A scripting language is a language that interprets scripts at runtime.
- The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application. Server side scripts are interpreted on the server while client side scripts are interpreted by the client application.
- PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages.
- The **development of a web application** is similar in many ways to that of any other software system. We have to find out what the users require, choose an appropriate software architecture, design and build the overall framework and create all the necessary components, all the while testing the evolving system against its technical and user expectations and adapting to changing requirements and circumstances.
- PHP is a widely used open source language that is specifically used for web application development and can be embedded within HTML. PHP is a server-side scripting language that is used in Web-based applications.
- PHP is a general-purpose programming language originally designed for web application development. PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is an open source, server side programming language. PHP is one of the most popular scripting languages of the last couple of years for developing web application.
- PHP is an interpreted language, i.e. there is no need for compilation. PHP files have extension ".php". PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- PHP is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- The basic architecture of a PHP web application and how the server handles the requests is shown in Fig. 1.2.

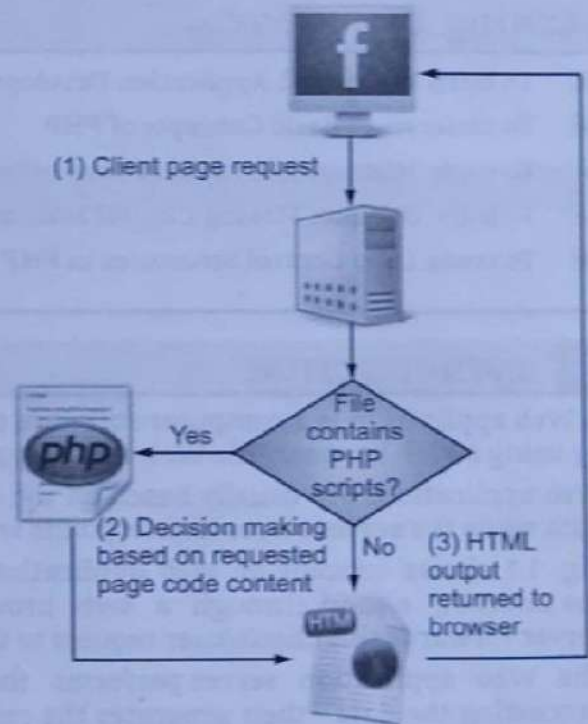


Fig. 1.2: Basic Architecture of PHP Web App

#### Features of PHP:

1. **Web Application Features:** PHP is a programming language support most commonly used Web application features like HTTP Cookie, Session, File Upload, etc.
2. **CLI (Command Line Interface):** PHP is commonly used to write Web applications to be integrated to Web servers using the Common Gateway Interface (CGI). But PHP can also be used to write standalone applications to be executed by the CLI (Command Line Interface).
3. **Built-in Modules:** The standard PHP build comes with many free and open source libraries included as built-in modules for calendar dates handling, FTP protocol support, XML processing, encryption and decryption, ODBC support, ZIP support, regular expression support, etc.
4. **Object-Oriented Programming (OOP):** PHP supports object-oriented programming features like object, classes, object references, private and protected member variables and methods, abstract and final classes and methods, constructors and destructors, interfaces, etc.

5. **Pre-Compilation:** PHP Web applications are usually deployed in source code, which will be interpreted on-the-fly when Web requests arrive to the server. But PHP Web applications can also be pre-compiled and deployed in executable format to speed-up Web requests response time.
6. **Real Time Access Monitoring:** PHP provides access logging by creating the summary of recent accesses for the user.
7. **File I/O:** PHP supports most commonly used file I/O features like local file and path management, remote resource access using Internet protocols like HTTP and FTP.

## 1.1 HISTORY, ADVANTAGES AND SYNTAX OF PHP

- In this section we study history, syntax, advantages and disadvantages of PHP.

### 1.1.1 History of PHP

- PHP was written in the C programming language by Rasmus Lerdorf in 1994 for use in monitoring his online resume and related personal information. For this reason, PHP originally stood for "Personal Home Page".
- Lerdorf combined PHP with his own Form Interpreter, releasing the combination publicly as PHP/FI (generally referred to as PHP 2.0) on June 8, 1995.
- Two programmers, Zeev Suraski and Andi Gutmans, rebuilt PHP's core, releasing the updated result as PHP/FI 2.0 in 1997.
- The acronym was formally changed to PHP: HyperText Preprocessor, at this time, this is an example of a recursive acronym (where the acronym itself is in its own definition).
- In 1998, PHP 3 was released, which was the first widely used version.
- PHP 4 was released in May 2000, with a new core, known as the Zend Engine 1.0. PHP 4 featured improved speed and reliability over PHP 3. In terms of features, PHP 4 added references, the Boolean type, COM support on Windows, output buffering, many new array functions, expanded object-oriented programming, inclusion of the PCRE library, and more. Maintenance releases of PHP 4 are still available, primarily for security updates.
- On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II. PHP 5 included new features such as improved support for object-oriented programming, the PHP Data Objects (PDO) extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements.
- On 2 November 2006, PHP 5.2 was released, with native JSON support.
- On 30 June 2009, PHP 5.3 was released, with Namespace support; late static bindings, jump label (limited goto), closures, PHP archives (phar), garbage collection for circular references, improved Windows support, sqlite3, mysqlnd as a replacement for libmysql as underlying library for the extensions that work with MySQL, fileinfo as a replacement for mime\_magic for better MIME support, the Internationalization extension, and deprecation of ereg extension.
- On 1 March 2012, PHP 5.4 was released with several improvements to existing features, performance and reduced memory requirements. Removed items: register\_globals, session\_register(), session\_unregister().
- On 20 June 2013, PHP 5.5 was released, with generators and finally blocks for exceptions handling support.
- On 28 August 2014, PHP 5.6 version is released which includes features like Constant expressions, Default character encoding, pgsql async support, an interactive debugger phpdbg and so on.
- On 3 December 2015 a new major PHP version was developed, which was numbered PHP 7.0 PHP version 7 comes with features like Null coalescing operator, Scalar type declarations, Return type declarations, Spaceship operator and many more.
- On 1 December 2016 PHP 7.1 version was released with features like Nullable types, Asynchronous signal handling, Class constant visibility, Symmetric array destructuring and so on.
- On 30 November 2017 PHP 7.2 version was released with features like Abstract method overriding, New object type etc.



- On 6 December 2018 PHP 7.3 version was released with features like Abstract method overriding, Parameter type widening, Flexible heredoc and nowdoc syntaxes and so on.
- The latest version of PHP is 7.4 released on 28 November 2019, offers to build applications that influences everything from the website and mobile to organizations and the cloud.

### 1.1.2 Advantages and Disadvantages of PHP

#### Advantages of PHP:

1. **Speed:** It is relative fast since it uses much system resource.
2. **Easy to Use/Simplicity:** It uses C like syntax, so for those who are familiar with C, it's very easy for them to pick up and to create a website.
3. **Stable:** Since it is maintained by many developers, so when bugs are found, it can be quickly fixed.
4. **Platform Independent/Portability:** Can be run on many platforms, including Windows, Linux and Mac, it's easy for users to find hosting service providers.
5. **Open Source:** PHP is open source and free of cost. It can be downloaded (without any fee) anywhere and readily available to use for the development of web applications.
6. **Built-in Database Connection Modules:** We can connect to database easily using PHP, since many websites are data/content driven, so we will use database frequently, this will largely reduce the development time of web apps.
7. **Powerful Library Support:** We can easily find functional modules we need such as PDF, Graph etc.
8. **Flexibility:** Because PHP is an embedded language, it is extremely flexible towards meeting the needs of the developer.
9. **Database Connectivity:** The PHP based application can easily be loaded and connected to the database. PHP support for many databases connectivity such as MySQL, Oracle, DB2, PostgreSQL etc.
10. **Speed-up Custom Web Application Development:** Nowadays, PHP programmers have to write web applications based on complex requirements.

#### Disadvantages of PHP:

1. **Security:** PHP is not that secure because of its open source, as the source code can be easily available.
2. **Weak Type:** Implicit conversion may surprise unwary programmers and lead to unexpected bugs. For example, the strings "1000" and "1e3" compare equal because they are implicitly cast to floating point numbers.
3. **Not Suitable for Large Web Applications:** PHP programs/codes are hard to maintain since it is not very modular. PHP unable to handle/manage large numbers of applications.
4. **Desktop Applications:** Not good to create desktop applications.
5. **Modification Problem:** PHP do not allow the change or modification in core behavior of the web applications.

### 1.1.3 Syntax of PHP

- A PHP script is executed on the server, and the plain HTML result is sent back to the browser.
- A PHP script can be placed anywhere in the document. A PHP programming script starts with `<?php` and ends with `?>`.

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

- The default file extension for PHP files is ".php". A PHP file normally contains HTML tags and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page.

**Example:** First PHP program.

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
    echo "Hello World!";
?>
</body>
</html>
```

**Output:**

```
My first PHP page
Hello World!
```

- PHP code can be included or embedded into web page in following ways:
  1. SGML style <? php code... ?>
  2. ASP style <% php code... %>
  3. Script style <script language = "php"> PHP Code... </script>

### 1.1.4 Step to Run Program using XAMPP Server

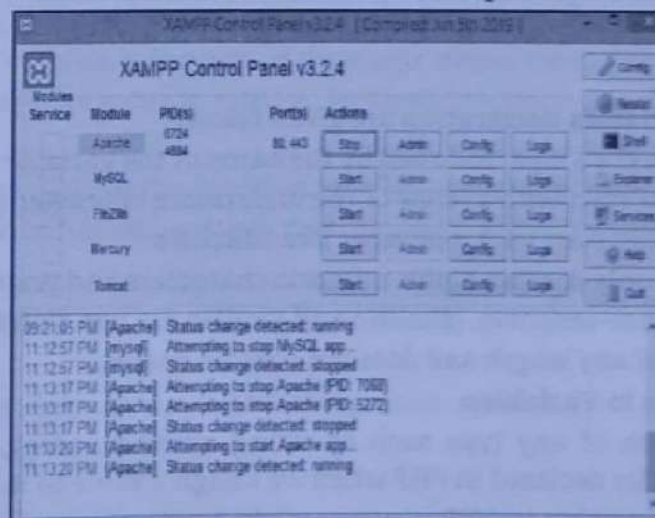
- XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). XAMPP is a software distribution which provides the Apache web server, MySQL database (actually MariaDB), PHP and Perl (as command-line executables and Apache modules) all in one package.
- AMPP server is available for Windows, MAC and Linux operating systems.
- Follow the following steps to run PHP program using XAMPP server:

**Step 1:** Download the XAMPP server from the internet,

(<https://www.apachefriends.org/download.html>).

**Step 2:** Install the XAMPP software.

**Step 3:** Open and click on the start button that is in front of Apache text.

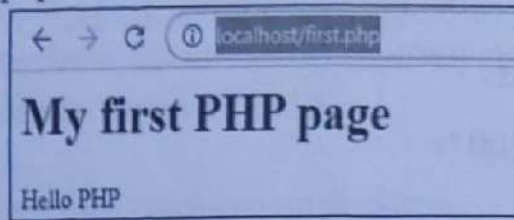


**Step 4:** Create php file in htdocs directory which is resides in xampp directory. [C:\xampp\htdocs].

**Example:** (first.php). The PHP echo statement is often used to output data to the screen.

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
    echo "Hello PHP";
?>
</body>
</html>
```

**Step 5:** Save first.php in htdocs directory then open your browser and then type `http://localhost/first.php` on address bar and press Enter key.



## 1.2 VARIABLES

- PHP variables are nothing but a named storage locations in the memory. A variable is a named container in a PHP script in which a data value can be stored.
- The stored value can be referenced using the variable's name and changed (varied) as the script proceeds/executes.
- Variables in PHP are identifiers prefixed with a dollar sign (\$). For example,

```
$name
$Age
$_Address
$MAXIMUM_IMPACT
```

- A variable may hold any type of value. There is no compile-time or runtime type checking on variables. We can store any type of value in the same variable.

For example,

```
$a = "Hello";
$a = 12;
$a = array(10, 20, 30);
```

### Variable Declaration:

- A variable is an identifier for a piece of data stored in memory during the program execution.
- A variable starts with the \$ sign, followed by the name of the variable.

**Syntax:** \$variable;

**For example:** \$PhoneNo;

- Some **rules for PHP variables declaration** are given below:
  1. A variable starts with the \$ sign, followed by the name of the variable like \$Sum.
  2. A variable name must start with a letter or the underscore character (\_).
  3. A variable name cannot start with a number like 10RollNo.
  4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9 and \_).
  5. Variable names are case-sensitive, (\$RollNo and \$rollNo are two different variables).
  6. PHP variable can be of any length and does not contain spaces.

### Defining/Assigning Values to Variables:

- A variable stores a value of any type such as string, number, array, object, resource and so on. A variable is automatically declared in PHP when we assign a value to it.
- Assigning a value to a variable in PHP is accomplished with the assignment operator (=), with the variable on the left-hand side and the expression to be evaluated on the right.

- **Syntax to define a variable** in PHP is: \$variable\_name=value;

**For example:** \$EmpId=10;

```
$StudentName="Vedant";
```

### Variable Variables:

- PHP allows us to use dynamic variable names, called variable variables. Variable variables are simply variables whose names are dynamically created by another variable's value.
- Sometimes, it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically. Value of existing variable is used as a name of new variable.

- A variable takes the value of a variable and treats that as the name of a variable.

```
$a = 'hello'; //hello is value of variable $a
```

```
$$a = 'PHP'; //$( $a ) is equals to $(hello)
```

```
echo $hello; // $hello is PHP i.e. #hello is new variable with value 'PHP'
```

#### Variable References:

- Variable reference is an alias (duplicate name) of existing variable.
- A variable reference points to the same value as the variable that references it. In PHP we can create reference to some variable.

For example,

```
$a = 5;
```

```
$b = &$a;
```

Here, \$b is the reference variable or \$b is an alias for the variable \$a. \$b is now another name for the value that is stored in \$a.

- Now, same value can be used by both the names.

```
echo $b;
```

```
$b = $b + 2;
```

```
echo $a;
```

#### Output:

7 (\$b is alias of \$a i.e. \$a and \$b are same variable).

- We can unset the variable by using unset function but the reference is still set.

```
unset($a);
```

```
echo $b; // Output: 5
```

#### Variable Scope:

- Scope of variable is an area or part of program in which it is accessible/visible. In PHP, variables can be declared anywhere in the script.
- Scope can be defined as, "the range of availability of a variable has to the program in which it is declared". The scope of a variable is the part of the script where the variable can be referenced/used.
- There are four types of variable scope in PHP i.e., local, global, static and function parameter.

##### 1. Local Scope:

- A variable which is declared inside the function is called as local variable.
- Local variable has access or life only within that function. Local Variable cannot be accessed from outside the function.

**Example:** For local scope.

```
<?php
    $a = 4; // global scope, $a is a global variable
    function assigna()
    {
        $a = 0; // local variable $a
        print "a inside function is $a.";
    }
    assigna();
    print "a outside of function is $a. ";
?>
```

#### Output:

a inside function is 0.

a outside of function is 4.

Functions can provide local scope. Unlike in other languages, in PHP we can't create a variable scope is a loop, conditional branch, or other type of block.

**Global Scope:**  
Variables declared outside function are by default global variables and can be accessed from any part of the program. A global variable can be accessed in any part of the program except inside the functions. Inside the function, these variables are accessed using the 'GLOBAL' keyword.

**Example:** For global scope.

```

x = 15;
y = 20;
function addit()
{
    GLOBAL $x;          // global variable;
    $x = 10;            // global variable;
    $x++;
    $y++;
    printf "x = $x & y = $y";
    addit();
}
// Output:
// x = 16 and y = 11

```

**Variable:**

Now that, when function ends then all the variables declared inside the function are destroyed. Static variables are those variables which can hold value when function called again.

Static variables are used only with the functions. These variables retain their values between subsequent calls to a function.

To declare a variable to be static simply by placing the keyword static in front of the variable. The initialization of this variable is done only for the first call to function and not after that.

**Example:** For static scope/variable.

```

function keep_track()
{
    static $count = 0;
    $count++;
    printf "count: %d\n", $count;
    printf "<br>";
}
// Output:
// count: 1
// count: 2
// count: 3

```

#### 4. Function Parameter:

- Function parameters are local, i.e. they are available only inside the function.
- Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable.

**Example:** For function parameter.

```
<?php
    // multiply a value by and return it to the caller
    function multiply ($value)
    {
        $value = $value * 4;
        return $value;
    }
    $retval = multiply (10);
    Print "Return value is $retval\n";
?>
```

**Output:**

Return value is 40

#### PHP Pre-defined Variables:

- PHP automatically have some variables called pre-defined variables available anywhere in the program. They are array variables and known as superglobals. These variables are `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, and `$_SERVER`, referred to as EGPCS.
- There is a setting in the configuration file (php.ini) called `register_globals`. The default value is OFF, and it restricts how we can access some predefined variables. `register_globals` determine whether or not to register the EGPCS variables as global variables.
- Regardless of the setting of the option `register_globals`, PHP creates following six global arrays that contain the EGPCS information as given below:
  1. **\$\_COOKIE:** This global array contains any cookie values passed as part of the request, where the keys of the array are the names of the cookies.
  2. **\$\_GET:** This global array contains any parameters that are part of a GET request, where the keys of the array are the names of the form parameters.
  3. **\$\_POST:** This global array contains any parameters that are part of a POST request, where the keys of the array are the names of the form parameters.
  4. **\$\_FILES:** This global array contains information about any uploaded files.
  5. **\$\_SERVER:** This global array contains useful information about the web server, as described in the next section.
  6. **\$\_ENV:** This global array contains the values of any environment variables, where the keys of the array are the names of the environment variables.
  7. **\$GLOBALS:** Contains a reference to every variable which is currently available within the global scope of the script. The keys of this array are the names of the global variables.
  8. **\$\_SERVER:** This is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server. There is no guarantee that every web server will provide any of these.
  9. **\$\_SESSION:** An associative array containing session variables available to the current script.

### 1.2.1 Data Types

- A data type is defined as, "a set of values and the allowable operations on those values". The data type determines the operations that we can perform on it.
- PHP supports eight primitive types as shown in Fig. 1.3.

- **Four Scalar Types:** (integer, float, string and Boolean).
- **Two Compound Types:** (array and object).
- **Two Special Types:** (resource and Null).

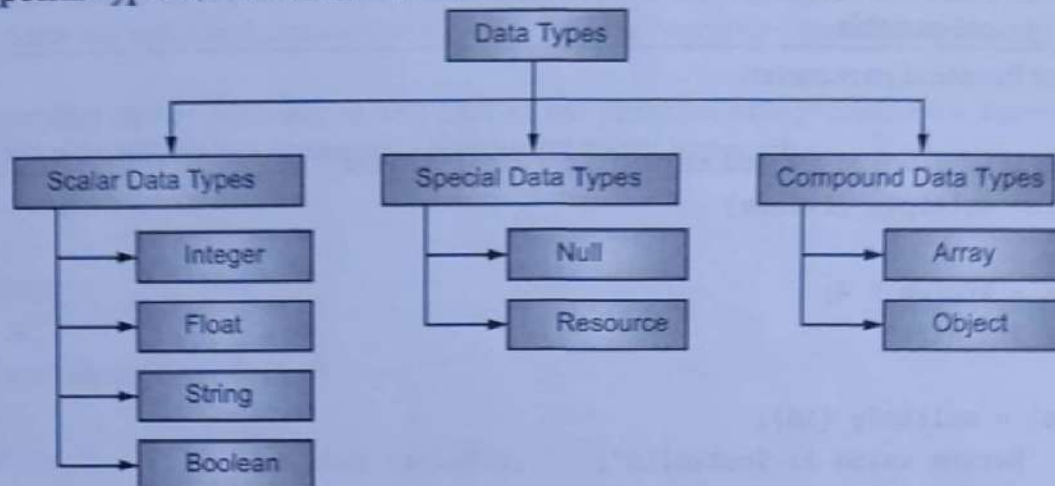


Fig. 1.3: Data Types in PHP

### 1. Integers:

- The integer data type is used to specify a numeric value without a fractional component. The range of integers in PHP is equivalent to the range of the long data type in C.
- On 32-bit platforms, integer values can range from -2,147,483,648 to +2,147,483,647.
- Integers can be written in decimal, octal or hexadecimal. If it is decimal, it is just the number. If it is octal, then number should precede with (zero) 0. If it is hexadecimal then precede with 0X.
- PHP does not support unsigned integers. Integer size can be determined using the constant PHP\_INT\_SIZE.
- In PHP, is\_int() or is\_integer() are used to test whether a value is an integer.

```

<?php
    if(is_int($a))
    {
        echo"Number is an integer";
    }
?>

```

### 2. Floating Point Numbers:

- Floating point numbers are real numbers, representing numeric values with decimal digits. Usually, this allows numbers between  $1.7E - 308$  and  $1.7E + 308$  with 15 digits of accuracy.
- PHP recognizes floating point numbers written in two different formats:
  - (i) **Common format:** 3.14, 0.25.
  - (ii) **Scientific format:** 17.0E-3 // 17.0\*10<sup>-3</sup>, or 0.017.
- Use the is\_float() function (or its is\_real() alias) to test whether a value is a floating point number:

```

if (is_float($x))
{
    // $x is a floating-point number
}

```

### 3. Strings:

- A string is a sequence of characters where a character is the same as a byte. PHP only supports a 256-character set.
- String literals are delimited by either single or double quotes.  
**Example:** For 'Good' and "Good" is same.

- Variables are expanded within double quotes not within single quote.

```
$a="Good";
echo "$a, morning \n";
echo '$a, morning';
```

**Output:**

```
Good, morning
$a, morning
```

- Use `is_string()` function to test whether the value is string or not.

```
if(is_string($x))
{
    // $x is a string
}
```

#### 4. Boolean:

- Boolean value can be either TRUE value or FALSE value. Both are case-insensitive.

For example,

```
<?php
    $x = True; // assign the value TRUE to $x
?>
```

- In PHP, `is_bool()` function is used to test whether value is Boolean or not.

```
$x = True;
if(is_bool($X))
{
    // $x is boolean;
}
```

- In PHP, the following values are false:

- The keyword false.
- The integer 0.
- The floating-point value 0.0.
- The empty string ("" ) and the string "0".
- An array with zero elements.
- An object with no values or functions.
- The NULL value.

#### 5. Arrays:

- An array stores group of values under single variable name. In PHP array is a collection of the different type of values.

- The values can be identified by position or some identifying name called as associative.

```
$a = array('A', 'B', 'C');
$b = array('First' => 'A',
          'Second' => 'B',
          'Third' => 'C');
```

\$a is indexed array and elements are recognized by index starting with 0.

```
i.e. $a[0]="A";
     $a[1]="B";
     $a[2]="C";
```

\$b is associative array in which key and value both are given.

```
i.e. $b['First'] = "A";
     $b['Second'] = "B";
```



```
$b['Third'] = "C";
```

foreach loop is most common in arrays.

```
i.e. foreach ($a1 as $value1)
{ echo "Hello, $a1 \n";
}
```

## 6. Objects:

- PHP supports Object Oriented Programming (OOP). Objects are the special instances of the classes that are created by user.
- Object is a term used in association with classes. A class is a definition of a structures that contains properties (variables) and methods (functions).
- Classes are defined with the 'class' keyword. Once a class is defined, any number of objects can be made from it with the 'new' keyword.
- Objects properties and methods can be accessed with the -> construct.

**Example:** For objects in PHP.

```
<?php
class Person
{
    public $name = ' ';
    function name ($newname)
    {
        $this -> name = $newname;
    }
}
$p = new Person();
$p -> name("Amar");
echo "Hello $p->name";
?>
```

### Output:

Hello Amar

- Use `is_object()` to test whether a value is an object.

```
if(is_object($x))
{
    // $x is an object
}
```

## 7. Resources:

- The resource is a special PHP data type that refers to external resource (e.g. file, image etc.) which is not part of the PHP native language.
- It is basically used for dealing with the outside world. Resources are created and used by special functions. For example, database connection function returns a resource which is used to identify that connection when you call the query and close functions.
- Their main benefit is that they're garbage collected when no longer in use. When the last reference to a resource value goes away, the extension that created the resource is called to free any memory, close any connection, etc. for that resource.

For example,

```
$result = database_connect();
database_query ($result);
$result="something"; // connection is closed.
```

- Use the `is_resource()` function to test whether a value is a resource.

```
if(is_resource($x))
```

Example	Name	Result
$-\$a$	Negation	Opposite of $\$a$ .
$\$a + \$b$	Addition	Sum of $\$a$ and $\$b$ .
$\$a - \$b$	Subtraction	Difference of $\$a$ and $\$b$ .
$\$a * \$b$	Multiplication	Product of $\$a$ and $\$b$ .
$\$a / \$b$	Division	Quotient of $\$a$ and $\$b$ .
$\$a \% \$b$	Modulus	Remainder of $\$a$ divided by $\$b$ .
$\$a ** \$b$	Exponentiation	Result of raising $\$a$ to the $\$b$ 'th power. Introduced in PHP 5.6.

- The division operator ("/") returns a float value unless the two operands are integers, (or strings that get converted to integers).

### String Concatenation Operator:

- The concatenation operator returns the concatenation of its right and left operands. Operands are first converted to strings, if necessary. For example:

```
<?php
    $a = "Hello ";
    $b = $a . "World!"; // now $b contains "Hello World!"
    echo $b;           // Display Hello World!
?>
```

### 2. Autoincrement and Autodecrement Operators:

- PHP supports C-style pre- and post-increment and decrement operators.
- The increment/decrement operators only affect numbers and strings. Arrays, objects and resources are not affected. Decrementing NULL values has no effect too, but incrementing them results in 1.

Example	Name	Effect
$++\$a$	Pre-increment	Increments $\$a$ by one, then returns $\$a$ .
$\$a++$	Post-increment	Returns $\$a$ , then increments $\$a$ by one.
$--\$a$	Pre-decrement	Decrements $\$a$ by one, then returns $\$a$ .
$\$a--$	Post-decrement	Returns $\$a$ , then decrements $\$a$ by one.

- These operators can be applied to strings as well as numbers. Incrementing an alphabetic character turns it into the next letter in the alphabet.

For example:

Incrementing "a" gives "b"

Incrementing "z" gives "aa"

Incrementing "spaz" gives "spba"

Incrementing "K9" gives "L0"

### 3. Comparison Operators:

- PHP provides comparison operators to compare two operands. A comparison operator returns a Boolean value, either true or false. If the comparison is truthful, the comparison operator returns true, otherwise it returns false.
- There are following comparison operators supported by PHP language:

Example	Name	Result
$\$a == \$b$	Equal	True if $\$a$ is equal to $\$b$ after type juggling.
$\$a === \$b$	Identical	True if $\$a$ is equal to $\$b$ , and they are of the same type.

contd....

$\$a \neq \$b$ or $\$a \diamond \$b$	Not equal	True if $\$a$ is not equal to $\$b$ after type juggling.
$\$a \neq \$b$	Not identical	True if $\$a$ is not equal to $\$b$ , or they are not of the same type.
$\$a < \$b$	Less than	True if $\$a$ is strictly less than $\$b$ .
$\$a > \$b$	Greater than	True if $\$a$ is strictly greater than $\$b$ .
$\$a \leq \$b$	Less than or equal to	True if $\$a$ is less than or equal to $\$b$ .
$\$a \geq \$b$	Greater than or equal to	True if $\$a$ is greater than or equal to $\$b$ .

#### 4. Logical Operators:

- Generally, logical operator used in decision making. Logical operators treat their operands as Boolean values and return a Boolean value.
- There are following logical operators supported by PHP language:

Example	Name	Result
$\$a \& \$b$ , $\$a$ and $\$b$	AND	True if both $\$a$ and $\$b$ are True.
$\$a \ \ \$b$ , $\$a$ or $\$b$	OR	True if either $\$a$ or $\$b$ is True.
$\$a \text{ xor } \$b$	XOR	True if either $\$a$ or $\$b$ is True, but not both.
$! \$a$	NOT	True if $\$a$ is not True.

#### 5. Assignment Operators:

- The basic assignment operator ( $=$ ) assigns a value to a variable. The lefthand operand is always a variable. The righthand operand can be any expression-any simple literal, variable, or complex expression. The righthand operand's value is stored in the variable named by the lefthand operand.
- In addition to the basic assignment operator, there are "combined operators" (or short hand assignment operator) for all of the binary arithmetic, array union and string operators that allow you to use a value in an expression and then set its value to the result of that expression.

For example:

```
<?php
$a = 3;
$a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
?>
```

- These assignment operators are plus-equals ( $+=$ ), Minus-equals ( $-=$ ), Divide-equals ( $/=$ ), Multiply-equals ( $*=$ ), Modulus-equals ( $\%=$ ), Bitwise-XOR-equals ( $\^=$ ), Bitwise-AND-equals ( $\&=$ ), Bitwise-OR-equals ( $\|=$ ) and Concatenate-equals ( $\.=$ ).

#### 6. Bitwise Operators:

- Bitwise operators perform operations on the binary representation of the operands. The following illustrates bitwise operators in PHP:

Example	Name	Result
$\$a$ and $\$b$	AND	If both bits are 1, the corresponding bit in the result is 1; otherwise, the corresponding bit is 0.
$\$a \ \ \$b$	OR (inclusive or)	If both bits are 0, the resulting bit is 0; otherwise, the resulting bit is 1.
$\$a \wedge \$b$	XOR (exclusive or)	If either of the bits in the pair, but not both, is 1, the resulting bit is 1; otherwise, the resulting bit is 0.

contd...

<code>~ \$a</code>	NOT	changes 1s to 0s and 0s to 1s in the binary representations of the operands.
<code>\$a &lt;&lt; \$b</code>	Shift Left	Shift the bits of \$a \$b steps to the left (each step means "multiply by two").
<code>\$a &gt;&gt; \$b</code>	Shift Right	Shift the bits of \$a \$b steps to the right (each step means "divide by two").

### 7. Casting Operators:

- Casting operator changes type of its operand by force. PHP casting operators are (int), (float), (string), (bool), (array), and (object).

For example:

```
$a = "5"; // type of $a is string
$b = (int) $a; // type of $b is integer
```

### 8. Miscellaneous Operators:

- (i) **Error Suppression (@):** This operator is also called as error control operator, works only on expression. When the operator is used with expression then any error messages that might be generated by that expression will be ignored.

- (ii) **Execution ('...'):** PHP supports one execution operator backticks (' '). Note that these are not single-quotes. PHP will attempt to execute the contents of the backticks as a shell command.

```
$output = `ls -l`; // shell command for long list of files.
echo $output; // Displays long list of files from present directory.
```

- (iii) **Conditional (?:):** It is also called as ternary operator.

**Syntax:** `expr1? expr2: expr3`

Meaning is, if `expr1` is true, execute `expr2` otherwise `expr3`.

**For example:**

```
<?php
$a = 10;
$b = 15;
$max = $a > $b? $a: $b;
$min = $a < $b? $a: $b;
echo "max = $max";
echo "min = $min";
?>
```

**Output:**

```
max = 15
min = 10
```

## 1.2.3 Operator Precedence and Associativity

- The precedence and associativity of operators are significant characteristics of a programming language. Operator precedence is a characteristic of operators that determines the order in which they evaluate the operands surrounding them. The associativity characteristics of an operator specifies how operations of the same precedence are evaluated as they are executed.
- Associativity can be performed in two directions, left-to-right or right-to-left. Left-to-right associativity means that the various operations making up the expression are evaluated from left to right.
- Operator Precedence is the order in which operators in an expression are evaluated. For example, in the expression `1 + 5 * 3`, the answer is 16 and not 18 because the multiplication ("\*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: `(1 + 5) * 3` evaluates to 18.

- When operators have equal precedence their associativity decides how the operators are grouped. For example "-" is left-associative, so  $1 - 2 - 3$  is grouped as  $(1 - 2) - 3$  and evaluates to  $-4$ . "=" on the other hand is right-associative, so  $\$a = \$b = \$c$  is grouped as  $\$a = (\$b = \$c)$ .
- Operators of equal precedence that are non-associative cannot be used next to each other, for example  $1 < 2 > 1$  is illegal in PHP. The expression  $1 <= 1 == 1$  on the other hand is legal, because the `==` operator has lesser precedence than the `<=` operator.
- The following table shows the operators with the highest precedence appear at the top of the table; those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	!, ++, --	Right-to-left
Multiplicative	*, /, %	Left-to-right
Additive	+, -	Left-to-right
Relational	<, <=, >, >=	Left-to-right
Equality	==, !=	Left-to-right
Logical AND	&and	Left-to-right
Logical OR		Left-to-right
Conditional	?:	Right-to-left
Assignment	=, +=, -=, *=, /=, %=	Right-to-left

### 1.2.4 Constants

- A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script.
- A constant is a value that cannot be changes in the PHP script, once the value has been assigned to the named variable. The value is available to the code only after it has been declared.
- A constant is case-sensitive by default. By convention, constant identifiers are always uppercase. To create a constant, use the `define()` function.
- Only single values i.e. scalars can be constants, like Boolean, integer, double, string etc. Value of constants are set using the `define()` function.

**Syntax:** `define("constant_Name", value);`

**For example,**

```
define('PI', 3.14);
echo PI;
```

- We can also use the function `constant()` to read a constant's value if we wish to obtain the constant's name dynamically.

## 1.3 DECISION MAKING CONTROL STATEMENTS

- The process of determining the order in which statements execute in a program is called decision making or flow control. A statement is code that performs a task.
- The control statements are used to control the flow of execution of the program. This execution order depends on the supplied data values and the conditional logic.
- PHP supports a number of traditional programming constructs for controlling the flow of execution of a program.
- Conditional statements allow a program to execute different piece of code depending on the condition such as `if-else` and `switch`.

### 1.3.1 if Statement

- The `if` statement allows us to make decision based on one or more conditions and execute a piece of code conditionally.

- The if statement in PHP contain statements that are only executed when the condition is satisfied means the condition is true. The if statement cannot do anything if the condition is false.
- The **syntax** of the if statement:

```
if(expression)
    statements
```

OR

```
if (condition)
{
    // put the code here;
}
```

- The following is an example of using the PHP if statement. We have \$x variable with value 11. In the if statement, we check if value of \$x is greater than 10 we display a message.

```
<?php
    $x = 11;
    if($x > 10)
    {
        echo '$x is greater than 10';
    }
```

- PHP if statement flow diagram is shown in Fig. 1.5.

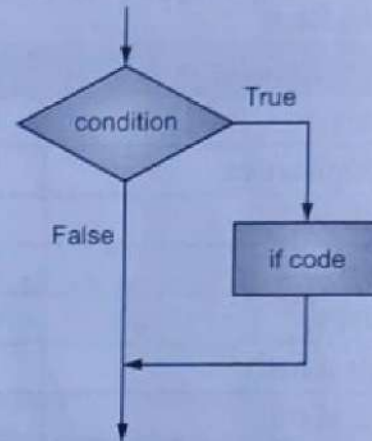


Fig. 1.5 : Flow diagram of if Statement

- If the expression evaluates to true, the code block inside the if statement is executed.

**For example:** for if statement.

```
<?php
    $a="Amar";
    $b="Akbar";
    if ($a==$b)
        echo "Both string are equal";
        echo "if block not executed";
?>
```

**Output:**

```
if block not executed
```

### 1.3.2 if-else Statement

- The if statement allows us to run a block of code if the condition evaluates to true. If the condition evaluates to false, the if statement skips the code block inside its body.
- The following illustrates the **syntax** of the if else statement:

```
if(expression)
    Statement...
else
    Statement...
```

OR

```
if (condition)
{
    // execute if condition evaluates to True...
}
else
{
    // execute if condition evaluates to False...
}
```

- The following example demonstrates how to use the PHP if else statement:

```
<?php
    $x = 2;
    if($x > 10){
        echo "$x is greater than 10";
    }
    else
```

```
{
    echo "$x is less than 10";
}
```

- PHP if-else statement flow diagram is shown in Fig. 1.6.

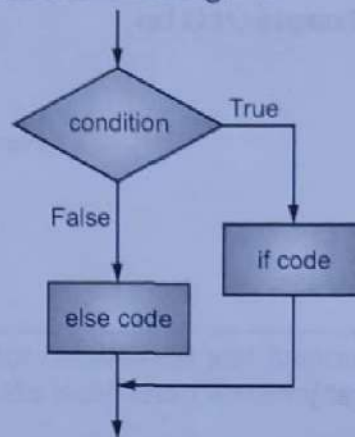


Fig. 1.6 : Flow diagram of if-else Statement

- We can enhance the if statement by adding else statement to run an alternative code block if the condition evaluates to false.

**Example:** For if-else statement.

```
<?php
    $var1="php";
    $var2="java";
    if ($var1 == $var2)
        echo "Both the strings are equal";
    else
        echo "Both the strings are not equal";
?>
```

**Output:**

Both the strings are not equal

**Example:**

```
<?php
    $marks=80;
    if($marks>= 60 )
    {
        echo"You are passed".<br>;
        echo"Your marks= ".$marks."<br>";
        echo"First division";
    }
    else
    {
        echo"Failed";
    }
?>
```

**Output:**

You are passed  
Your marks= 80  
First division

**Example:**

```

<html>
<head>
  <title>PHP Decision Making Example</title>
</head>
<body>
<?php
  $num1 = 5;
  $num2 = 10;
  $num3 = 15;
  if($num1>$num2)
  {
    echo "5 is greater than 10";
  }
  if($num3>$num2)
  {
    echo "15 is greater than 10";
  }
?>
</body>
</html>

```

**Output:**

15 is greater than 10

- If we have more than one Boolean condition to test, we can use the if elseif statement.

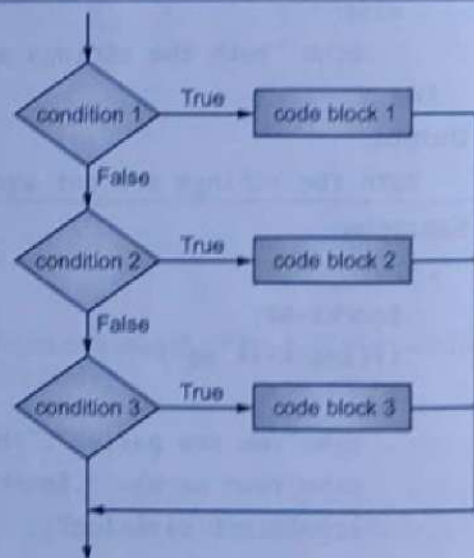
**Syntax:**

```

if(condition)
{
  //code block of if branch to be executed
}elseif(condition2)
{
  //code block of elseif branch to be executed
}elseif(condition3)
  //code block of elseif branch to be executed
}else
{
  //code block of else branch to be executed
}

```

- PHP if elseif statement flow diagram is shown in Fig. 1.7.



**Fig. 1.7 : Flow diagram of if-elseif Statement**

**Example: For if-elseif statement**

```

<?php
  $x = 20;
  if($x > 0)
  {
    echo "$x is greater than zero"
  }

```



```

else if($x == 0)
{
    echo "$x is zero";
}
else
{
    echo "$x is less than zero";
}
?>

```

**Output:**

20 is greater than zero.

- PHP also provide an alternative for conditional and looping control structure. In place of opening brace colon (:) is used and to close the block endif keyword is used (in this case).

**Syntax:**

```

if(condition);
    //code block to be executed if condition is true
    //...
else
    // code block to be executed if condition is false
    // ...
endif;

```

**Example:**

```

<?php
$a=7
if ($a == 5):
    echo "a equals 5";
    echo "...";
elseif ($a == 6):
    echo "a equals 6";
    echo "!!!";
else:
    echo "$ais neither 5 nor 6";
endif;
?>

```

**Output:**

a is neither 5 nor 6

**1.3.3 Nested if Statement**

- When one if statement contains another if statement then such type of structure is known as nested if. Nested if structure also helps in multi-way decision making where one condition depends on other.
- It has the following **form/syntax**:

```

if (condition 1)
{
    if (condition 2)      /* nested if */
    {
        Statement(s);
    }
}

```

```
}  
else  
{  
    Statement(s);  
}  
}  
else  
{  
    Statement(s);  
}
```

**Example:** For nested if statement.

```
<!DOCTYPE html>  
<html>  
<body>  
<?php  
    $a=10;  
    $b=20;  
    if($a==10)  
    {  
        if($b==20)  
        {  
            echo "The addition is: " . ($a+$b);  
        }  
    }  
?>  
</body>  
</html>
```

**Output:**

The addition is: 30

### 1.3.4 switch Statement

- Consider the case where value of a single variable may determine one of a number of different choices (e.g., the variable holds the username and we want to do something different for each user). The switch statement is designed for just this situation.
- Switch statement is used to compare the value with multiple cases or values. All statements in a matching case are executed upto the first break keyword.
- If none match and 'default' is given, all statements following the default keyword are executed up to the first break keyword.

**Syntax:**

```
switch(variable)  
{  
    case value1:  
        // code block 1  
        break;  
    case value2:  
        // code block 2  
        break;
```

```

    default:
    // default code block
}

```

- **Alternative Syntax**

```

switch(variable):
    case value1:
    // code block 1
    break;
    case value2:
    // code block 2
    break;
    default:
    // default code block
endswitch;

```

- PHP switch statement flow diagram is shown in Fig. 1.8.

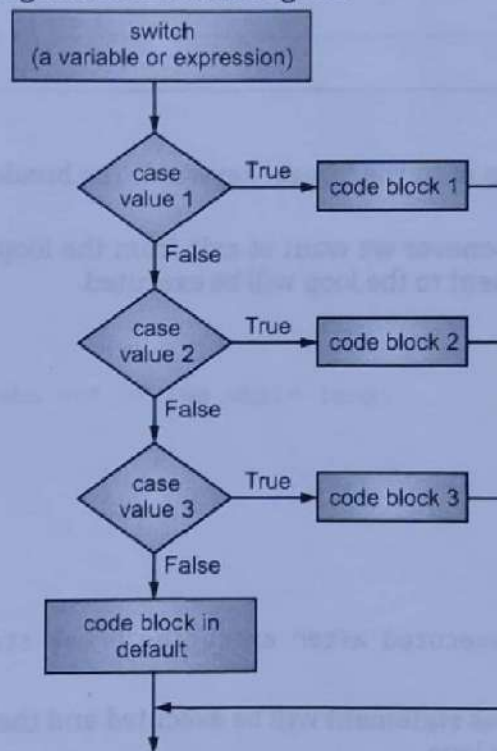


Fig. 1.8 : Flow diagram of switch Statement

**Example:** For switch statement.

```

<?php
$var=date("D");
switch($var)
{
case "Sun":
    echo "It is Sunday.";
    break;
case "Mon":
    echo "It is Monday.";
    break;
case "Tue":
    echo "It is Tuesday.";

```

```

break;
case "Wed":
    echo "It is Wednesday.";
    break;
case "Thu":
    echo "It is Thursday.";
    break;
case "Fri":
    echo "It is Friday.";
    break;
case "Sat":
    echo "It is Saturday.";
    break;
default:
    echo "Something wrong";
?>
    
```

**Output:**

It is monday

**1.3.5 break Statement**

- We can prematurely exit a loop with the 'break' keyword. The break statement is situated inside the statement block.
- It gives us full control and whenever we want to exit from the loop we can come out. After coming out of a loop immediate statement to the loop will be executed.

**Syntax:**

```

for(...)
{
    // -----
    if(condition)
    {
        break;
    }
    // ----- /* Never executed after executing break statement*/
}
    
```

- If the condition is true, the break statement will be executed and the loop will be break. It will skip all the remaining statement of the loop.

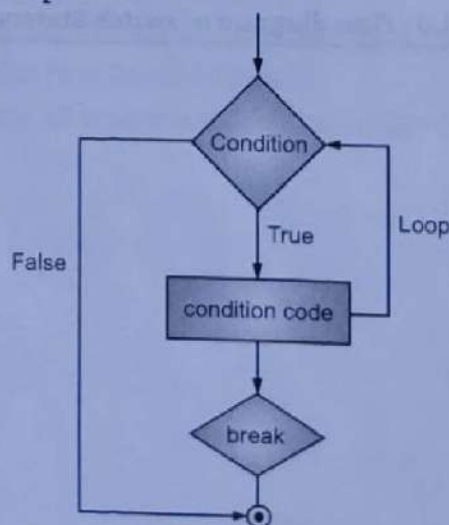


Fig. 1.9: Flow Diagram for break Statement

- In the following code, \$i never reach a value of 6, because the loop is stopped once it reaches 5.

**Example:** For break statement.

```
<?php
    $i=1;
    while($i<=10)
    {
        echo $i . " ";
        if($i == 5)
            break;
        $i++;
    }
?>
```

**Output:**

1 2 3 4 5

- Optionally, we can put a number after the break keyword, indicating how many levels of loop structures to break out of. In this way, a statement buried deep in nested loops can break out of the outermost loop.

**Example:**

```
<?php
    $i = 1;
    while ($i <= 10)
    {
        $j = 1;
        while ($j <= 10)
        {
            if ($j == 5)
                break 2; // breaks out of two while loops
            $j++;
        }
        $i++;
    }
    echo $i;
    echo "<br>";
    echo $j;
?>
```

**Output:**

1  
5

### 1.3.6 continue Statement

- The continue statement forces the next iteration of the loop skipping the remaining code of the loop.
- The continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

**Syntax:**

```
for(initialisation;condition;increment/decrement)
{
    ...
    if(True Condition) //Continues Loop with the Next Value
        continue;
}
```

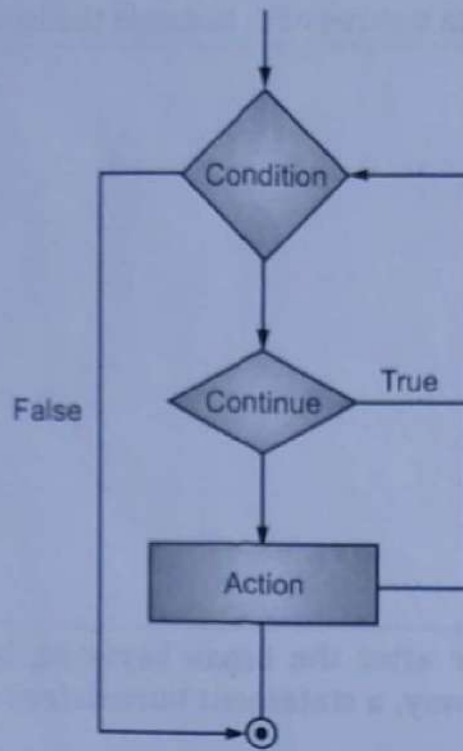


Fig. 1.10: Flow Diagram for continue Statement

Following program displays only odd numbers from the array.

**Example:** For continue statement.

```
$a = array(1,2,3,4,5);
```

```
foreach($a as $value)
```

```
if($value%2 == 0)
```

```
continue;
```

```
echo $value . " ";
```

```
3 5
```

## LOOP CONTROL STRUCTURES

PHP is used to execute a statement or a block of statements, multiple times until and unless a condition is met. This helps the user to save both time and effort of writing the same code multiple times.

Loops in PHP are used to execute the same block of code a specified number of times. PHP has the following four loop types:

**for:** Loops through a block of code if and as long as a specified condition is true.

**while:** Loops through a block of code once and then repeats the loop as long as special condition is true.

### 1.4.1 while Loop

- The while loop is the simplest loop statement in PHP. While loop will execute block of code until certain condition is met.

#### Syntax:

```
while(expression)
{
    //code block to be executed
}
```

- The while loop statement checks the expression at the beginning of each iteration. If the expression evaluates to true, the code block inside the curly braces is executed. If the expression evaluates to false, the loop exits.

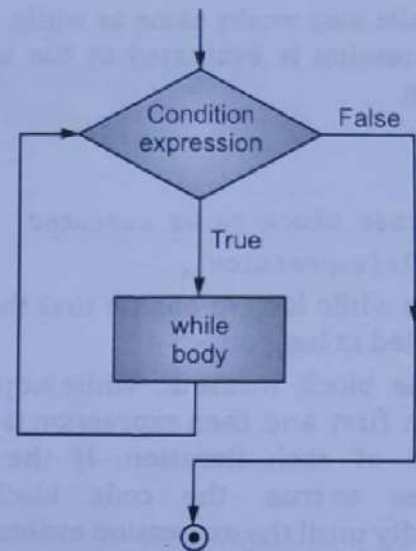


Fig. 1.11: Flow Diagram for while Loop

**Example:** Program to displays number from 1 to 10.

```
<?php
    $i=1;
    while($i<=10)
    {
        echo $i . " ";
        $i++;
    }
?>
```

#### Output:

1 2 3 4 5 6 7 8 9 10

- The **alternative syntax** for while statement is:

```
while(expression):
    statement;
...;
endwhile;
```

**Example:** for while loop.

```
<?php
    $i=1;
    while($i<=10):
    echo $i . " ";
        $i++;
    endwhile;
?>
```

#### Output:

1 2 3 4 5 6 7 8 9 10

### 1.4.2 do while Loop

- A do while loop works same as while, except that the expression is evaluated at the end of each iteration.

**Syntax:**

```
do {
    //code block to be executed
} while(expression);
```

- Use a do while loop to ensure that the loop body is executed at least once.
- The code block inside do while loop statement executes first and then expression is checked at the end of each iteration. If the expression evaluates to true, the code block executes repeatedly until the expression evaluates to false.

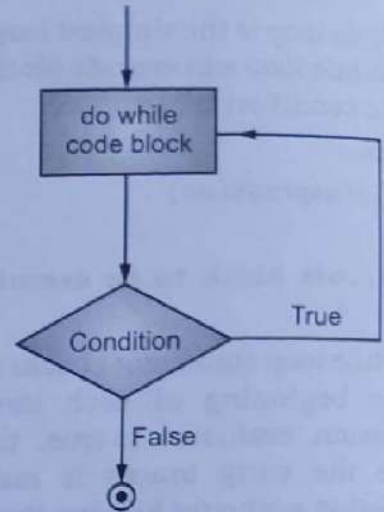


Fig. 1.12: Flow Diagram for do while Loop

**Example:** Program displays number from 1 to 10.

```
<?php
    $i=1;
    do {
        echo $i . " ";
        $i++;
    }while($i<=10);
?>
```

### 1.4.3 for Loop

- The for loop is same as while loop but it is shorter and easier to use.

**Syntax:**

```
for(init_expr; condition_expr; increment_expr)
{
    //code block to be executed
}
```

- At the beginning, the counter initialization occurs only once. Each time through the loop, the expression condition is tested. If it is true, the body of the loop is executed; if it is false, the loop ends. The expression increment/decrement is evaluated after the loop body runs.

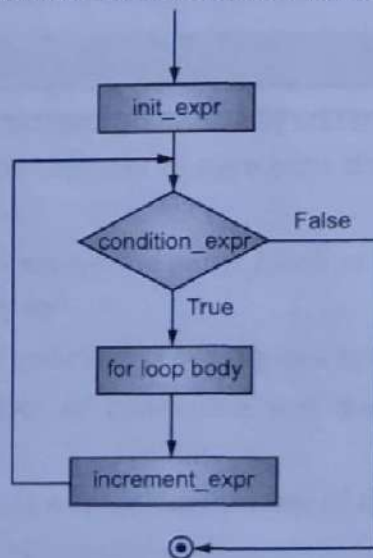


Fig. 1.13: Flow Diagram of 'for' Loop



**Example:** Program displays number from 1 to 10.

```
<?php
  for ($i=1; $i<=10; $i++)
  {
    echo $i . " ";
  }
?>
```

- The **alternative syntax** for 'for' loop:  
for (initialization; condition; increment):  
statement;  
...;  
endfor;

**Example:**

```
<?php
  for ($i=1; $i<=10; $i++):
  echo $i . " ";
  endfor;
?>
```

#### 1.4.4 foreach Loop

- The foreach construct provides an easy way to iterate over arrays. foreach works only on arrays and objects, and will issue an error when we try to use it on a variable with a different data type or an uninitialized variable.
- PHP provides the foreach loop statement that allows you to iterate over elements of an array or public properties of an object.
- There are two **syntax**:

```
foreach (array_expression as $value) statement      OR      foreach (array_expression as $key => $value) statement
```

- The first form loops over the array given by array\_expression. On each iteration, the value of the current element is assigned to \$value and the internal array pointer is advanced by one (so on the next iteration, we will be looking at the next element).
- The second form will additionally assign the current element's key to the \$key variable on each iteration.

**Example:** For foreach loop.

```
<?php
  $arr = array(1, 2, 3, 4);
  foreach ($arr as $value)
  {
    echo $value . " ";
  }
?>
```

**Output:**

1 2 3 4

- The **alternative syntax** for foreach loop is:  
foreach (array\_expression as \$value):  
// PHP code here...  
end foreach;

```
{
    // $x is a resource
}
```

### 8. Null:

- This data type is having only one value denoted by the keyword NULL.
- The null value represents a variable that has no value. A variable is considered to be null if:
  - (i) it has been assigned the constant NULL.
  - (ii) it has not been set to any value yet.
  - (iii) it has been unset().
- `is_null()` function is used to test whether a value is null or not null.

## 1.2.2 Expressions and Operators

- Processing data is accomplished in PHP programming, as it is in other programming languages, by way of operators and expressions.
- Operators are the symbols that tell PHP what operations to perform, and expressions are the individual sets of variables and operators that make a result when processing is complete.

### 1.2.2.1 Expressions

- Expressions are any code that evaluates to a value. The assignment of a value to a variable is an expression in itself, although we tend to think of expressions as similar to equations, (like  $Z = X + Y$ , where  $X + Y$  is the expression we have in mind). Therefore,  $X = 5$  is an expression, because it evaluates to then value 5.
- An expression is a piece of code that evaluates to a value. A value a number, a string of text, or a Boolean. An expression is a combination of values, variables, operators, and functions that results in a value.
- The simplest expressions are literal values, variables and complex expressions can be formed using simple expressions and operators.

### 1.2.2.2 Operators

- An operator is a symbol that manipulates one or more values, usually producing a new value in the process. The PHP operators are used to perform the operations in PHP.
- Operators are special symbols that perform some specific operations using the operands and operator as shown in Fig. 1.4.

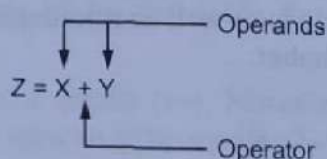


Fig. 1.4: Concept of Operator and Operand

- Operators indicate the operation to be carried out on operands, while operands are the values going to be operated.
- Categories of operators are as follows:
  1. **Unary Operators:** Operates on single operand.
  2. **Binary Operators:** Which take two operands and produces output/result value.
  3. **Conditional Operator (Ternary Operator):** Which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression
- Let see various operators in PHP in detail.
- 1. **Arithmetic Operators:**
  - The arithmetic operators require numeric values. And non-numeric values are converted automatically to numeric values.
  - There are following arithmetic operators supported by PHP language:

## Practice Questions

1. What is web application?
2. How to develop a web application?
3. What is PHP?
4. Enlist features of PHP.
5. What is variable? How to declare it? Explain with example.
6. Explain constants in PHP with example.
7. What are the decision statements used by PHP?
8. Describe loops in PHP.
9. What is operator? Which operators used by PHP? Explain with example.
10. Define the term expressions with example.
11. Describe data types in PHP in detail.
12. Explain variable scope in PHP.
13. Write the difference between break and continue statement of PHP with example.
14. Explain any two control statements with syntax and example.
15. Describe syntax of PHP with example.
16. Find output:

```
<?php
$a= "9 Lives." -1;
var_dump($a);
?>
```

17. Find the output.

```
<? php
$ str = "welcome";
echo 'You are $ str';
? >
```

18. Write program to find a area of rectangle.

# 2...

# Arrays, Functions and Graphics

## Chapter Outcomes...

- Manipulate the given type of arrays to get the desired result.
- Apply implode, explode functions on the given array.
- Apply the given string functions on the character array.
- Scale the given image using graphics concepts/functions.

## Learning Objectives...

- To understand Concepts of Array
- To learn Types of Arrays
- To study Traversing Arrays and Extracting Data from Arrays
- To understand Basic Concepts of Functions in PHP
- To learn String and String Functions
- To understand Basic Graphics Concepts in PHP

## 2.0 INTRODUCTION

- A function is a named block of code that is designed to perform a specific task. Once, a function is defined, we can reuse it without copying and pasting a code block again and again.
- A function may accept one or more arguments, which are the values that we pass to the function. A function may return a value so that the calling script can communicate with it.
- In PHP, a string is a sequence of characters. PHP also support graphics programming with functions that create, open and manipulate graphics.
- An array in PHP is a collection of key/value pairs. PHP arrays are a collection of variables, which stores multiple values of different data type at a time.

## 2.1 CREATING AND MANIPULATING ARRAY

- An array is a collection of data values, organized as an ordered collection of key-value pairs. In PHP array is used to store multiple values in single variable.
- Commonly used terms in array are explained below and shown in Fig. 2.1.

1. **Element:** Element of an array is the items it contains. An array can contain one or more elements.
2. **Value:** Each element contains one value.
3. **Key or Index:** Key or index of an array is a unique number or a string that is associated with each value of an element.
4. **Length:** The length of an array is the number of elements it contains.

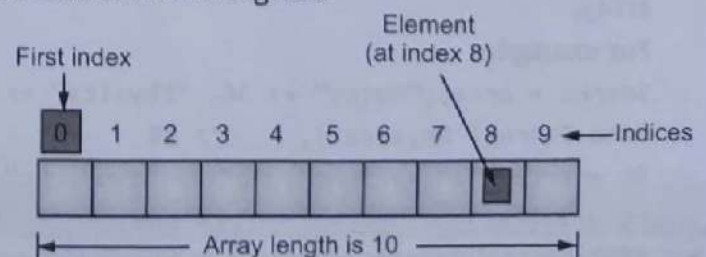


Fig. 2.1: Element of Array

[2.1]

**Creating an Array:**

- In PHP, the `array()` function is used to create an array. An array takes any number of comma-separated key => value pairs as arguments.

**Syntax:**

```
$array_name = array
(
key1 => value1,
key2 => value2,
key3 => value3,
...
);
```

- The key can either be an integer or a string. The value can be of any type.

**For example:**

```
<?php
(
$month = array
    0 => "January";
    1 => "February";
);
?>
```

**2.1.1 Types of Arrays**

- In PHP, there are three types of arrays namely, Indexed array, Associative array and Multidimensional array.

**2.1.1.1 Indexed Array**

- The keys of an indexed array are integers beginning at 0. Indexed arrays are used when identification of array elements are by their position.
- An array having only integer keys is typically referred to as an indexed array. Indexed arrays can store numbers, strings and any object but their index will be represented by numbers.

**For example:**

```
$city = array( 0 => "Pune", 1 => "Mumbai", 2 => "Delhi", 3 => "Chennai");
echo $city[1]; // Mumbai
```

- Indexed array can also be created without keys. In this case the key will be started from 0.

**For example:**

```
$city = array("Pune", "Mumbai", "Delhi", "Chennai");
echo $city[3]; // Chennai
```

**2.1.1.2 Associative Array**

- An associative array has strings as keys. An array having string keys is typically called an associative array.

**For example:**

```
$marks = array("Maths" => 36, "Physics" => 28, "Chemistry" => 30);
echo $marks['Physics']; // 28
$v = array("a" => "one", "b" => "two", "c" => "three");
echo $v['a']; // one
```

- PHP internally stores all arrays as associative arrays, so the only difference between associative and indexed arrays is what the keys happen to be. In both cases, the keys are unique-that is, we can't have two elements with the same key, regardless of whether the key is a string or an integer.

### 2.1.1.3 Multi-Dimensional Arrays

- In a multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.
- Values in the multi-dimensional array are accessed using multiple index.

```
$row0=array(1, 2, 3);
```

```
$row1=array(4, 5, 6);
```

```
$row2=array(7, 8, 9);
```

```
$multi=array($row0, $row1, $row2);
```

```
$val=$multi[2][0]; // 2nd row, 0th column and $val=7
```

**Example:** For multi-dimensional array.

```
<?php
    $marks = array
    (
        "Amar" => array
        (
            "physics" => 35,
            "maths" => 30,
            "chemistry" => 39
        ),
        "Kiran" => array
        (
            "physics" => 30,
            "maths" => 32,
            "chemistry" => 29
        ),
        "Deepa" => array
        (
            "physics" => 31,
            "maths" => 22,
            "chemistry" => 39
        )
    );
    /* Accessing multi-dimensional array values */
    echo "Marks for Amar in physics: " ;
    echo $marks['Amar']['physics'] . "<br />";
    echo "Marks for Kiran in maths: ";
    echo $marks['Kiran']['maths'] . "<br />";
    echo "Marks for Deepa in chemistry: " ;
    echo $marks['Deepa']['chemistry'] . "<br />";
?>
```

**Output:**

```
Marks for Amar in physics: 35
```

```
Marks for Kiran in maths: 32
```

```
Marks for Deepa in chemistry: 39
```

## 2.2 EXTRACTING DATA FROM ARRAYS

- PHP provides two functions `extract()` and `compact()`, that convert between arrays and variables. We can use the `extract()` function to extract data from arrays and store it in variables. The `compact()` function creates an array from variables and their values.

- The `extract()` function automatically creates local variables from an array. The indexes of the array elements are the variable names.

**Example:**

```
<?php
    $ice_cream["good"] = "mango"
    $ice_cream["better"] = "vanilla"
    $ice_cream["best"] = "butterscotch"
?>
```

Now we can we extract to create variables whose names will be taken from the keys in the array, and those variables will be assigned the values in the array.

```
<html>
    <head>
        <title> Extracting variables from arrays </title>
    </head>
    <body>
    <h1>Extracting variables from arrays</h1>
    <?php
        $ice_cream["good"] = "mango";
        $ice_cream["better"] = "vanilla";
        $ice_cream["best"] = "butterscotch";
        extract($ice_cream);
        echo "\$good = $good<br>";
        echo "\$better = $better<br>";
        echo "\$best = $best<br>";
    ?>
</body>
</html>
```

**Output:**

```
Extracting variables from arrays
$good = mango
$better = vanilla
$best = butterscotch
```

- The `compact()` function creates an array from variables and their values.

**Example:** For `compact()` in PHP.

```
<?php
    $n1 = 10;
    $n2 = 20;
    $n3 = 30;
    $arr = array("n1", "n2", "n3");
    $output = compact($arr);
    print_r($output);
?>
```

**Output:**

```
Array ([n1] => 10 [n2] => 20 [n3] => 30)
```

- The `list()` function is an inbuilt function in PHP which is used to assign array values to multiple variables at a time.

**2.2.1 Implode**

- We use the `implode()` function to convert an array into a string. The `implode` function in PHP is used to "join elements of an array with a string".

- It creates a string from an array of smaller strings. The implode() function returns a string from the elements of an array.
- Syntax:** string implode (string \$glue, array \$pieces)
- This function join array elements with a 'glue' string. The glue contains the information about defaults to an empty string while pieces is the array of strings to implode.
  - It returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.

**Example:** For implode().

```
<?php
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(", ", $array);
echo $comma_separated;
?>
```

**Output:**

```
lastname, email, phone
```

- join() is an alias of implode() function.

### 2.2.2 Explode

- It breaks a string into smaller parts and stores it in an array. To convert a string into an array, we use the explode() function.
- The explode() function splits the string based on the specified delimiter and returns an array that contains elements, which are substrings produced by the splitting operation.
- In short, the explode function is used to "split a string by a specified string into pieces i.e. it breaks a string into an array".

**Syntax:** array explode (string \$delimiter, string \$str [, int \$limit ])

- Returns an array of strings, each of which is a substring of 'str' formed by splitting it on boundaries formed by the string 'delimiter'.
- If 'limit' is set and positive, the returned array will contain a maximum of 'limit' elements with the last element containing the rest of 'str'.
- If the 'limit' parameter is negative, all components except the last - 'limit' are returned. If the 'limit' parameter is zero, then this is treated as 1.

**Example:** For explode().

```
<?php
$str = 'one|two|three|four';
$arr = explode('|', $str);
print_r($arr);
echo "<br>";
$arr = explode('|', $str, 2);
print_r($arr);
echo "<br>";
$arr = explode('|', $str, -1);
print_r($arr);
?>
```

**Output:**

```
Array ([0] => one [1] => two [2] => three [3] => four)
Array ([0] => one [1] =>two|three|four)
Array ([0] => one [1] => two [2] => three)
```



- Since, in the second case the limit is two, hence the string will be divided into two parts. In the third case since the limit is negative so except the component 'four', remaining components will be returned.
- The explode() function breaks a string into an array, but the implode function returns a string from the elements of an array.

### 2.2.3 Array Flip

- The array\_flip() function flips/exchanges all keys with their associated values in an array.
- PHP array\_flip() this function very useful when we have a big/large array, and we want to know if a given value is in the array.

**Syntax:** array\_flip(array)

**Example:** For array flip().

```
<?php
$a1 = array("a"=>"red", "b"=>"green", "c"=>"blue", "d"=>"yellow");
$result = array_flip($a1);
print_r($result);
?>
```

**Output:**

```
Array ([red] => a [green] => b [blue] => c [yellow] => d)
```

## 2.3 TRAVERSING ARRAYS

- Traversing an array means to visit each and every element of array using a looping structure and iterator functions.
- There are several ways to traverse arrays in PHP.

### 1. Using foreach Construct:

- PHP provides a very special kind of looping statement called foreach that allows us to loop over arrays. The foreach statement only works with arrays and objects.

**Example:** For foreach construct to traversing arrays.

```
$a = array('aaa', 'bbb', 'ccc');
foreach($a as $value)
{
    echo "$value\n";
}
```

**Output:**

```
aaa
bbb
ccc
```

- In the above foreach loop, the loop will be executed once for each element of \$a, i.e. three times. And for each iteration \$value will store the current element.

**Example:**

```
$a = array('one' => 1, 'two' => 2, 'three' => 3);
foreach ($A as $k => $v)
{
    echo "$k is $v \n";
}
```

**Output:**

```
one is 1
two is 2
three is 3
```

- In this case the key for each element is stored in \$k and the corresponding value is stored in \$v. The foreach operates on copy of array, not on array itself.
- 2. Using for Loop:**
- The for loop operates on the array itself, not on a copy of the array.

**Example:** For 'for' construct to traversing arrays

```
$a = array(1, 2, 3, 4);
for($i=0; $i<count($a); $i++)
{
    echo $A[i] . "<br>";
}
```

**Output:**

```
1
2
3
4
```

### Iterator Functions:

- Every PHP array keeps track of the current element. The pointer to the current element is known as the iterator. PHP has functions to set, move, reset this iterator.
- The iterator functions are:
  1. **current()**: Returns the currently pointed element.
  2. **reset()**: Moves the iterator to the first element in the array and returns it.
  3. **next()**: Moves the iterator to the next element in the array and returns it.
  4. **prev()**: Moves the iterator to the previous element in the array and returns it.
  5. **end()**: Moves the iterator to the last element in the array and returns it.
  6. **each()**: Returns the key and value of the current element as an array and moves the iterator to the next element in the array.
  7. **key()**: Returns the key of the current element.

**Example:** For interator functions.

```
<?php
    $transport = array('foot', 'bike', 'car', 'plane');
    $mode = current($transport); // $mode = 'foot';
    $mode = next($transport);    // $mode = 'bike';
    $mode = current($transport); // $mode = 'bike';
    $mode = prev($transport);    // $mode = 'foot';
    $mode = end($transport);     // $mode = 'plane';
    $mode = current($transport); // $mode = 'plane';
?>
```

- After executing each() the iterator moves to the next element.

```
<?php
    $a = array(10, 20, 30, 40, 50);
    reset($a);
    while (list($key, $val) = each($a))
    {
        echo "$key => $val<br>";
    }
?>
```

**Output:**

```
0 => 10
1 => 20
2 => 30
3 => 40
4 => 50
```

**Calling a Function for each Array Element:**

- The `array_walk()` function apply a user defined function to each element of an array.
- **Syntax:** `bool array_walk (array &$arr, callable $function_name)`
- This function takes two parameters, first is the input array and second is the user defined function name.
- The user defined function takes two arguments, first contains arr's value and second contains arr's key. Returns True on success or False on failure.

**Example:**

```
<?php
function print_row($v, $k)
{
    echo "$v $k <br>";
}
$a = array(10, 20, 30, 40);
array_walk($a, 'print_row');
?>
```

**Output:**

```
10 0
20 1
30 2
40 3
```

- In the above program the 'print\_row' function will be called four times (4) i.e. for each elements of the array \$a. The 'print\_row' function then displays the values along with the keys.

**Reducing an Array:**

- The `array_reduce()` function apply a user defined function to each element of an array, so as to reduce the array to a single value.
- **Syntax:** `mixed array_reduce(array $array, callable $callback [, mixed $initial = NULL ])`
- The function takes two arguments namely, the running total, and the current value being processed. It should return the new running total.

**Example:** For reducing array.

```
<?php
function add($sum, $value)
{
    $sum += $value;
    return $sum;
}
$n = array(2, 3, 5, 7);
$total = array_reduce($n, 'add');
echo $total;
?>
```

**Output:**

```
17
```

- The function 'add' will be called for each element, i.e. 4 times. The function then finds the sum and returns it.
- If the optional initial is available, it will be used at the beginning of the process.  

```
$total = array_reduce($n, 'add', 10);
cho $total; // 27 (i.e. 10 + 17)
```

#### Searching for Values:

- The `in_array()` function searches if a value exists in an array or not.  
**Syntax:** `bool in_array (mixed $to_find, array $input [, bool $strict = FALSE])`
- The `in_array()` function returns true or false, depending on whether the element 'to\_find' is in the array 'input' or not.

**Example:** For searching values.

```
<?php
    $os = array("Mac", "NT", "Irix", "Linux");
    if (in_array("Irix", $os))
    {
        echo "Got Irix";
    }
    if (in_array("mac", $os))
    {
        echo "Got mac";
    }
?>
```

- The second condition is false because `in_array()` is case-sensitive, so the program above will display:  
Got Irix
- If the third parameter `strict` is set to `True` then the `in_array()` function will also check the types of the \$value.

**Example:**

```
<?php
    $a = array(2, 3, "4", "5");
    if (in_array('3', $a, true))
    {
        echo "'3' found with strict check\n";
    }
    if (in_array('4', $a, true))
    {
        echo "'4' found with strict check\n";
    }
?>
```

**Output:**

```
'4' found with strict check
```

- In the above program the type of '3' which we are searching is string but in the array 'a', 3 is integer. Hence first condition is false.

#### array\_search():

- The `array_search()` function search an array for a value and returns the key.

**Syntax:** `mixed array_search (mixed $to_find, array $input [, bool $strict = FALSE])`

- The `in_array()` function returns the key of the element 'to\_find' if it is found in the array 'input', otherwise returns False.

**Example:** For array search.

```
<?php
    $a=array("a"=>"5", "b"=>5, "c"=>"5");
    echo array_search(5,$a,true);
?>
```

**Output:**

b

## 2.4 FUNCTION AND ITS TYPES

- A function is an independent named block of code that is defined to perform a specific task. A function can be called from another part of the program, and may or may not return a value. Code inside a function are not executed until the function is called.
- There are two types of functions i.e., built-in functions and user defined functions.
  1. **Built-in Functions** are also called as pre-defined functions or library functions that perform a wide range of operations and they can be retrieved by the programmer directly. PHP is very rich in terms of built-in functions.
  2. **User Defined Functions** are defined by user.
- Functions are useful because they contribute to rapid, reliable, error-reducing coding. A function can be called many times in a page but it is compiled only once.
- As a function is written only once and removes the need to write the codes for the same task every time it requires, it reduces number of bugs. As functions separate codes that perform a specific task, it improves readability.

**Advantages of using Functions:**

1. PHP Functions reduces the memory and complexity of the program.
2. Functions make PHP script modular - by using functions, a big script is divided into many functions that are easier to build, test and maintain.
3. Functions mainly helps in code re-usability.
4. PHP functions reduces the bugs and saves the time of programmer.
5. Information hiding is possible by functions.

### 2.4.1 Defining and Calling a Function

- We already know that a function is a named block of code that performs a specific task, in this section we discuss defining and calling functions.

#### 2.4.1.1 Defining a Function

- A function is a self-contained block of code that performs a specific task. A function is created in a PHP script simply by stating its name, after the 'function' keyword, followed by parentheses and curly brackets (braces { }) containing the statements to be executed each time that function gets called.
- To define a function following **syntax** is used:

```
function [&] function_name([parameter[, ...]])
{
    statement list
}
```
- A function starts with the keyword function and is followed by the function name. [&] is optional and if it is used then function returns reference.
- A function name can be any string that begins with a letter or underscore ( \_ ) followed by zero or more letters, underscores and numbers.

- Function names are case-insensitive. It means count(), Count() and COUNT() refer to the same function. By convention, PHP built-in functions are in lowercase.

**Example:** For function.

```
<?php
function writeMsg()
{
    echo "Hello world!";
}
writeMsg();           // call the function
?>
```

**Output:**

Hello world!

### 2.4.1.2 Calling a Function

- Calling a function is simple - just write the name of the function followed by a pair of parentheses.
- We can call a function by using following **syntax**:  
`$some_value = function_name([parameter, ...]);`
- After calling a function, the statements inside it are executed. After the function execution is completed, the program control returns to the point where the function was called.
- Fig. 2.2 shows how caller and receiver functions work together.

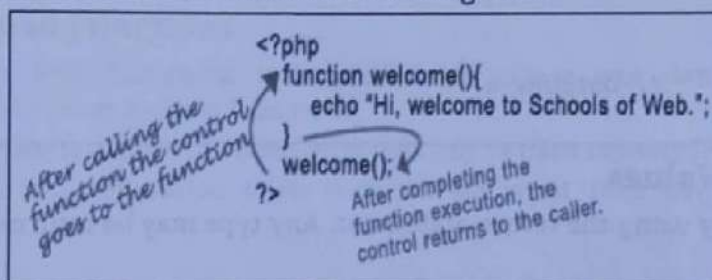


Fig. 2.2: How Caller and Receiver Functions Work Together

**Example:** For calling a function.

```
<?php
Function welcome()
{
    echo "Hi, Welcome to PHP.";
}
welcome(); // Function welcome() is called here.
?>
```

**Output:**

Hi, Welcome to PHP.

- **Explanation of Above Program:** The function welcome() is defined. In this stage nothing happens. When the function is called, the interpreter finds it and enters into it, executes the statement(s) inside it. Here, it prints the sentence "Hi, Welcome PHP."

### 2.4.1.3 Function Arguments

- Information can be passed to functions through arguments. Inside the brackets () after the function name, we specify one or more parameters (or arguments).
- If a function accepts more than one parameter, each parameter has to be separated by a comma(.). The arguments are evaluated from left to right.

**Example:**

```
function strcat($left, $right)
{
    $combined_string = $left.$right;
    echo $combined_string;
}
```

- PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are also supported.
- By default, function arguments are passed by value (so that if the value of the argument within the function is changed, it does not get changed outside of the function). To allow a function to modify its arguments, they must be passed by reference.
- When an argument to a function passed by reference, always prepend by ampersand (&) to the argument name in the function definition.

**Example:** For function with arguments.

```
<?php
function doubler(&$value)
{
    $value = $value * 2;
}
$a = 3;
doubler($a);
echo $a;           // Outputs 6
?>
```

**2.4.1.4 Returning Values**

- Values are returned by using the return statement. Any type may be returned, including arrays and objects.
- A function typically processes the arguments and returns a value to the caller, which may be another function or script. To return a value from a function, we use the return statement.
- When PHP encounters the return statement, it terminates the function immediately and returns the value back.

**Example:**

```
<?php
function square($num)
{
    return $num * $num;
}
echo square(4); // Outputs 16
?>
```

- A function can not return multiple values, but similar results can be obtained by returning an array.

**Example:** For function returning values.

```
<?php
function day_name()
{
    $day1 = "Monday";
    $day2 = "Tuesday";
    $day3 = "Wednesday";
    return array($day1, $day2, $day3);
}
```

```

    }
    $days = day_name();
    echo $days[0] . " " . $days[1] . " " . $days[2];
?>

```

**Output:**

Monday Tuesday Wednesday

- Function `day_name()` creates an array that has three elements in it and returns to its caller. When the function is called, the function returns the array to the caller assigns it in the `$days` variable. The next line prints the values of the array elements.

**2.4.1.5 Function Body**

- The code inside the curly brace `{ }` is a function body. We can put PHP code, HTML code or mixed PHP and HTML code inside its body.
- For example, the following function displays the header of a web page:

```

function display_header($header)
{
    echo "<h1>" . $header . "</h1>";
}

```

**2.4.2 Function Types**

- In this section we study various types of functions in PHP.

**2.4.2.1 User Defined Functions**

- Besides the built-in PHP functions, it is possible to create our own functions, according to requirements known as user defined functions.
- A user defined function is a block of statements that can be used repeatedly in a program.
- A user-defined function declaration starts with the key word 'function' followed by name as the function.

**Syntax:**

```

function functionName()
{
    code to be executed;
}

```

**Example:** For user defined functions.

```

<?php
function writeMsg()
{
    echo "Hello world!";
}
writeMsg();
?>

```

**Output:**

Hello world!

**2.4.2.2 Variable Function**

- PHP supports the concept of variable functions means that we can call function based on value of a variable. If a variable name has round parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it.
- The variable function does not work with `echo()`, `print()`, `unset()`, `isset()`. PHP allows us to store a name of a function in a string variable and use the variable to call the function.



- Variable functions are useful in cases that we want to execute functions dynamically at run-time.
- The following example demonstrates how to use variable functions.

```
<?php
    Function find_max($a,$b)
    {
        return $a > $b? $a: $b;
    }
    Function find_min($a,$b)
    {
        return $a < $b? $a: $b;
    }
    $f = 'find_max';
    echo $f(10,20);    // Outputs 20
    $f = 'find_min';
    echo $f(10,20);    // Outputs 10
?>
```

- First, we defined two functions namely, find\_min() and find\_max(). Then, we called those function based on the value of string variable \$f. Notice that we need to put parentheses after the variable in order to call the function specified by the value of the variable.

### 2.4.2.3 Anonymous Function (Lambda Function)

- We can define a function that has no specified name. We call this function is an anonymous function or a closure. Also called as lambda function. We often use anonymous functions as values of callback parameters.
- The anonymous function is created using the create\_function(). It takes two parameters. First parameter is the parameter list for anonymous functions and second parameter contains the actual code of the function. The function name is randomly generated and returned.

```
$f_name = create_function(args_string, code_string);
```

**Example:** For lambada function.

```
<?php
    $add = create_function('$a,$b', 'return($a+$b);');
    $r = $add(2,3);
    echo $r;
?>
```

**Output:**

5

## 2.5 OPERATIONS ON STRING AND STRING FUNCTIONS

- PHP string is a sequence of characters i.e., used to store and manipulate text. String is one of the data types supported by PHP. In this section we study string operations and functions.

### Concept of String:

- A "string" of text can be stored in a variable in much the same way as a numeric value but the assignment must surround the string with quote marks ("..." or '...') to denote its beginning and end like "Hello world!".
- Strings are very useful to store names, passwords, address, and credit-card number and so on. For that reason, PHP has large number of functions for working with strings.
- There are three different ways to write string in PHP as explained below:

#### 1. Single Quoted String:

- In this method, string is enclosed with single quotation mark ('...').

**For example:** 'Hello World' 'Fred' 'Pune' etc.

- The limitation of single quoted string is that variables are not interpolated.

**Example:** For single quoted string.

```
<?php
$name = 'Amar';
$str = 'Hello $name'; // Single-quoted string
echo $str;
?>
```

**Output:**

Hello \$name

- In the above output the value of variable \$name is not printed.

## 2. Double-Quoted String:

- It is the most commonly used quote to express string. Here characters are enclosed with double quotation marks ("...").  
For example: "Hello World", "fred", "Pune".
- PHP interpreter interprets variables and special characters inside double quotes. In the following example, the above example is re-written using double quotes.

**Example:** For double quoted string.

```
<?php
$name = 'Amar';
$str = "Hello $name"; // Double-quoted string
echo $str;
?>
```

**Output:**

Hello Amar

## 3. Here Documents (heredoc):

- Single quoted and double-quoted strings allows string in single line. To write multiline string into a program 'heredoc' is used.
- **Rules of using heredoc:**
  - heredoc syntax begins with three less than signs (<<<) followed by identifier (a user defined name). The identifier can be any combination of letters, numbers, underscores but the first character must be a letter or an underscore.
  - The string begins in the next line and goes as long as it requires.
  - After the string, the same identifier that is defined after the (<<<) signs in the first line should be placed at the end. Nothing can be added in this last line except a semicolon after the identifier and it is optional.
  - Space before and after <<< is essential.

**Syntax:**

```
$var_name = <<< identifier
    string statement
    string statements
    identifier;
```

**Example:** For heredoc string.

```
<?php
$str = <<< EOD
```

<code>\t</code>	tab
<code>\\</code>	Backslash
<code>\\$</code>	Dollar sign
<code>\{</code>	left brace
<code>\}</code>	right brace
<code>\[</code>	left square bracket
<code>\]</code>	right square bracket
<code>\o</code>	through <code>\777</code> ASCII character represented by octal value.
<code>\xo</code>	through <code>\XFF</code> ASCII character represented by hex value.

## 2.5.1 String Functions/String Operations

- In this section we study various string operations in PHP.

### 2.5.1.1 str\_word\_count()

- The `str_word_count()` is in-built function of PHP. It is used to return information about words used in a string or counts the number of words in a string.

**Syntax:** `str_word_count(string, return, char)`

**Parameters:**

string specify the string to check.

return specify the return value of the function. It is an optional.

**Values:**

0 is by default and returns number of words found

1 returns an array with the words.

2 returns value is the actual word.

char specify characters. It is an optional.

**Example:**

```
<?php
    $str="PHP JAVA";
    echo "Your string is:".$str;
    echo "<br>";
    echo "The total word is: ".str_word_count($str);
?>
```

**Output:**

Your string is: PHP JAVA

The total word is: 2

### 2.5.1.2 strlen()

- The `strlen()` function is predefined function of PHP. It is used to find the length of string or returns the length of a string including all the whitespaces and special character.

**Syntax:** `strlen(string);`

**Example:**

```
<?php
    $str = 'Java';
    echo "Your String is:".$str;
    echo "<br>";
    echo "The length of String is: ".strlen($str);
?>
```

**Output:**

Your String is:Java

The length of String is:4

**2.5.1.3 strrev()**

- This function takes a string and returns a *reversed copy* of it.

**Syntax:** string strrev(string \$string)

**Example:**

```
<?php
    echo strrev("Hello World!");
?>
```

**Output:**

!dlroWolleH

**2.5.1.4 strpos()**

- Find the position of the first occurrence of a substring in a string.  
**Syntax:** mixed strpos(string \$str, mixed \$find[, int \$offset = 0])
- Find the numeric position of the first occurrence of 'find' in the 'str' string. Mixed means any PHP data type. If 'find' is not a string, it is converted to an integer and applied as the ordinal value of a character. 'offset' specifies that, search will start this number of characters counted from the beginning of the string.
- The function returns the position of where the 'find' exists relative to the beginning of the 'str' string (independent of offset). Also note that string positions start at 0 and not 1. Returns False if the 'find' was not found.

**Example:**

```
<?php
    echo strpos("I am a PHP programmer", "am");
    echo "<br>";
    echo strpos("I am a PHP programmer", "am", 5);
?>
```

**Output:**

2  
16

- In the above program the position (index) of character 'I' is 0, hence it displays position of 'am' is 2. In the second case, searching starts from 5<sup>th</sup> character, hence 'am' is at position 16.

**2.5.1.5 strrpos()**

- The syntax of this function is same as strpos() function. This function finds the last occurrence of substring in the main string.

**Example:**

```
<?php
    echo strrpos("I am a PHP programmer", "am");
    echo "<br>";
    echo strrpos("I am a PHP programmer", "am", 5);
?>
```

**Output:**

16  
16

**2.5.1.6 str\_replace()**

- The str\_replace() function is a case-sensitive, built-in function of PHP which replaces some character of the string with other characters.

- It is used to replace all the occurrences of the search string with the replacement string.

**Syntax:** `str_replace($search, $replace, $string, $count)`

**Parameters:**

The `$search` parameter contains the value which is going to be searched for the replacement in the `$string`.

The `$replace` parameter holds that value which will replace the `$search` value in the `$string`.

The `$string` is an array or string in which search and replace value is searched and replaced.

The `$count` is the last and optional parameter. This variable stores the total number of replacement performed on a string `$string`.

**Example:**

```
<?php
$string = "Hi everyone!";
$search = 'Hi';
$replace = 'Hello';
echo '<b>'. "String before replacement:". '</br></b>';
echo $string. '</br>';
$newstr = str_replace($search, $replace, $string, $count);
echo '<b>'. "New replaced string is:". '</br></b>';
echo $newstr. '</br>';
echo 'Number of replacement = '.$count;
?>
```

**Output:**

```
String before replacement:
Hi everyone!
New replaced string is:
Hello everyone!
Number of replacement = 1
```

### 2.5.1.7 ucwords()

- The `ucwords()` is an in-built function of PHP, which is used to convert the first character of each word to uppercase in a string.

**Syntax:** `ucwords($string, $separator)`

**Parameters:**

The `$string` (required) is a mandatory parameter of this function, which specifies the input string that needs to be converted.

The `$separator` (optional) is an optional parameter of this function, which contains the words separator characters. It specifies a character that uses a separator for the words in the input string.

**Example:**

```
<?php
$input_str = "hello, my name is ravi.";
echo ucwords($input_str);
$input_str1 = "hellomy name is ravi.";
echo ucwords($input_str1, "|");
?>
```

**Output:**

```
Hello, My Name Is Ravi.
Hello|My|Name|Is|Ravi.
```

### 2.5.1.8 strtoupper()

- The `strtoupper()` is one of the most popular functions of PHP, which is widely used to convert the string into uppercase.

**Syntax:** `strtoupper($string)`

**Example:**

```
<?php
$input_str = "all is well";
$result_str = strtoupper($input_str);
echo $result_str;
?>
```

**Output:**

ALL IS WELL

### 2.5.1.9 strtolower()

- The `strtolower()` is one of the most popular functions of PHP, which is widely used to convert the string into lowercase.

**Syntax:** `strtolower($string)`

**Example:**

```
<?php
$input_str = "ALL IS WELL";
$result_str = strtolower($input_str);
echo $result_str;
?>
```

**Output:**

all is well

### 2.5.1.10 strcmp()

- `strcmp()` is a string comparison function in PHP. It is a built-in function of PHP, which is case sensitive, means it treats capital and the small case separately.
  - This function compares two strings and tells whether a string is greater, less, or equal to another string.
- Syntax:** `strcmp($str1, $str2);`
- This function returns integer value randomly based on the comparison.
    - Return 0 returns 0 if both strings are equal, i.e.,  $\$str1 = \$str2$
    - Return < 0 returns negative value if string1 is less than string2, i.e.,  $\$str1 < \$str2$
    - Return > 0 returns positive value if string1 is greater than string 2, i.e.,  $\$str1 > \$str2$

**Example**

```
<?php
$str1 = "hello php";
$str2 = "hello php";
echo strcmp($str1, $str2). " because both strings are equal. ";
?>
```

**Output:**

0 because both strings are equal.

## 2.6 BASIC GRAPHICS CONCEPTS

- PHP support basic computer graphics. An image is a rectangle of pixels of various colors.
- Colors are identified by their position in the palette, an array of colors. Each entry in the palette has three separate color values one for Red, one for Green, and one for Blue. Each value ranges from 0 (this color not present) to 255 (this color at full intensity).
- Image files are rarely a straightforward dump of the pixels and the palette. Instead, various file formats (GIF, JPEG, PNG, etc.) have been created that attempt to compress the data somewhat to make smaller files.

### 2.6.1 Creating Images

- Before creating images in PHP, we need to understand some basic image - related concepts. Computers usually create colors using a color theory model called the RGB model.
- RGB stands for Red, Green, and Blue, the three basic colors that are combined to create the colors that we see on the computer display (screen).
- Computers typically work with two types of images namely raster and vector. Raster images (also known as bitmap images) are made up of pixel data; in a 20 - pixel - wide by 20 - pixel - high color image there are 400 individual pixels making up the image, and each of these pixels has its own RGB color value.
- The vector image uses mathematical equations to describe the shapes that make up the image. The SVG (Scalable Vector Graphics) format is a good example of a vector image.
- Vector images are great for diagrams that include lines, curves, and blocks of color, but are not suitable for photographs or images.
- The image functions that PHP uses are based on the GD image library that is developed by Tom Boutell ([www.boutell.com](http://www.boutell.com)). PHP's GD image functions let us work with four main raster image formats JPEG, PNG and GIF for desktop Web browsers and WBMP images for mobile browsers.
- The `imagecreate()` function is used to create a new image. It is preferred to use `imagecreatetruecolor()` to create an image instead of `imagecreate()`.
- This is because the image processing occurs on the highest quality image possible which can be created using `imagecreatetruecolor()`.

**Syntax:** `imagecreate($width, $height)`

**Parameters:**

`width` is width of image.

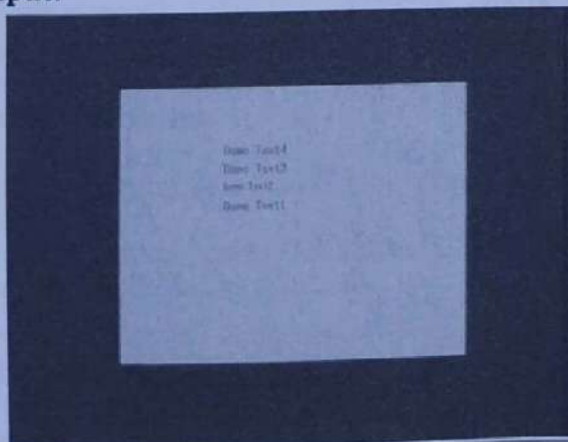
`height` is height of image.

`return:` `imagecreate()` function returns an image resource identifier on success or False on errors.

**Example:**

```
<?php
    $img = imagecreate(500, 300);
    $bgcolor = imagecolorallocate($img, 150, 200, 180);
    $fontcolor = imagecolorallocate($img, 120, 60, 200);
    imagestring($img, 12, 150, 120, "Demo Text1", $fontcolor);
    imagestring($img, 3, 150, 100, "Demo Text2", $fontcolor);
    imagestring($img, 9, 150, 80, "Demo Text3", $fontcolor);
    imagestring($img, 12, 150, 60, "Demo Text4", $fontcolor);
    header("Content-Type: image/png");
    imagepng($img);
    imagedestroy($img);
?>
```

**Output:**



- Use the `imagecolorallocate()` function to create the color. This function takes four arguments: the image resource created by `imagecreate()`, and the red, green, and blue components of the color that we like to create.

## 2.6.2 Images with Text

- The `imagefttext()` function is an inbuilt function in PHP which is used to write text to the image using TrueType fonts.

### Syntax:

```
imagefttext (resource $image, float $size, float $angle, int$x, int$y, int$color,
            string $fontfile, string $text) : array
```

### Parameters:

Image is a image resource, returned by one of the image creation functions, such as `imagecreatetruecolor()`.

size is the font size in points.

The angle in degrees, with 0 degrees being left-to-right reading text. Higher values represent a counter-clockwise rotation. For example, a value of 90 would result in bottom-to-top reading text.

x is the coordinates given by x and y will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the `imagestring()`, where x and y define the upper-left corner of the first character. For example, "top left" is 0, 0.

y is the y-ordinate. This sets the position of the fonts baseline, not the very bottom of the character.

color is the color index. Using the negative of a color index has the effect of turning off antialiasing.

fontfile is the path to the TrueType font we wish to use.

Text is the text string in UTF-8 encoding.

### Example:

```
<?php
// Set the content-type
header('Content-Type: image/png');
// Create the image
$im = imagecreatetruecolor(400, 30);
// Create some colors
$white = imagecolorallocate($im, 255, 255, 255);
$grey = imagecolorallocate($im, 128, 128, 128);
$black = imagecolorallocate($im, 0, 0, 0);
imagefilledrectangle($im, 0, 0, 399, 29, $white);
// The text to draw
$text = 'Testing...';
// Replace path by your own font path
$font = 'arial.ttf';
// Add some shadow to the text
imagefttext($im, 20, 0, 11, 21, $grey, $font, $text);
// Add the text
imagefttext($im, 20, 0, 10, 20, $black, $font, $text);
// Using imagepng() results in clearer text compared with imagejpeg()
imagepng($im);
imagedestroy($im);
?>
```

### Output:

## 2.6.3 Scaling Images

- The `imagescale()` function is an inbuilt function in PHP which is used to scale an image using the given new width and height.

**Syntax:** `imagescale($image, $new_width, $new_height = -1, $mode = IMG_BILINEAR_FIXED)`

### Parameters:

\$image is returned by one of the image creation functions, such as `imagecreatetruecolor()`. It is used to create size of image.



`$new_width` parameter holds the width to scale the image.

`$new_height` parameter holds the height to scale the image. If the value of this parameter is negative or omitted then the aspect ratio of image will preserve.

`$mode` parameter holds the mode. The value of this parameter is one of `IMG_NEAREST_NEIGHBOUR`, `IMG_BILINEAR_FIXED`, `IMG_BICUBIC`, `IMG_BICUBIC_FIXED` or anything else.

**Example:** For scaling an image.

```
<?php
// Assign image file to variable
$image_name= 'https://i2.cdn.turner.com/money/dam/assets/150901123238-google-new-
logo-780x439.png';
// Load image file
$image = imagecreatefrompng($image_name);
// Use imagescale() function to scale the image
$img = imagescale( $image, 500, 400 );
// Output image in the browser
header("Content-type: image/png");
imagepng($img);
?>
```

**Output:**



## 2.6.4 Creation of PDF Document

- FPDF is a PHP class which allows generating PDF (Portable Document Format) files with pure PHP that is to say without using the PDFlib library. F from FPDF stands for Free, we may use it for any kind of usage and modify it to suit we needs.
- FPDF has following features:
  1. Choice of measure unit, page format and margins.
  2. Page header and footer management.
  3. Automatic page break.
  4. Automatic line break and text justification.
  5. Image support, (JPEG, PNG and GIF).
  6. Colors and Links.
  7. TrueType, Type 1 and encoding support.
  8. Page compression.
- The FPDF library is free and can be downloaded from the official website's (<http://www.fpdf.org/>) download section. The download package contains all necessary files, along with some tutorials on how to use it.
- We must download and extract the FPDF package in the folder where the PHP file with the code is located to run this example.

**Example:** PHP Script to create a PDF file.

**createpdf.php**

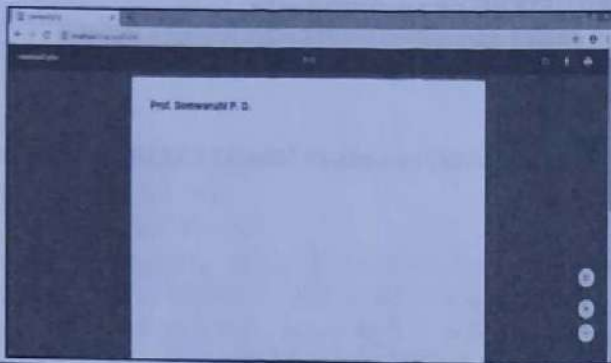
```
<?Php
require('fpdf.php');
$pdf = new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
```

```
$pdf->Cell(80,10,'Prof. Somwanshi P. D.');
```

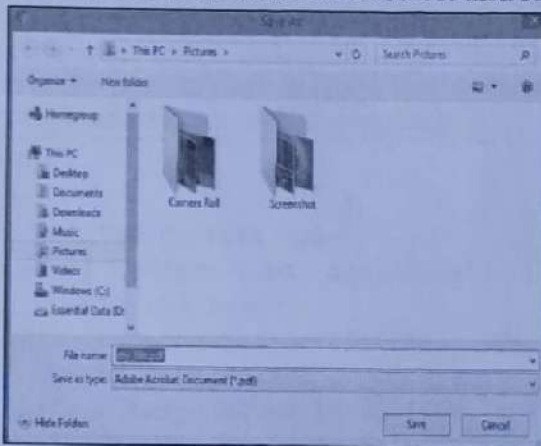
```
$pdf->Output('my_file.pdf','I'); // Send to browser and display
```

```
?>
```

**Output:**



- If once PDF file is created download it and save as "my\_file.pdf".



## Practice Questions

1. What is an array?
2. How to create an array? Explain with example.
3. What is string?
4. What is function?
5. How to extract data from array? Explain with example.
6. Enlist types of arrays. Describe with example.
7. Write the difference between index and associative array with example.
8. How to add image in PHP? Explain with example.
9. Define the terms:  
(i) Array (ii) String (iii) Function.
10. How to create a function? Explain with example.
11. State advantages of functions in PHP.
12. Compare string and array.
13. With the help of example describe traversing of array.
14. How to convert arrays into variables and vice-versa.
15. Which operations performed on string in PHP?
16. Enlist four string related function with syntax and example.
17. Explain the following types of functions with example:  
(i) User defined functions (ii) Variable function (iii) Anonymous function.
18. How to create PDF in PHP? Describe with example.
19. How to scaling an image in PHP?
20. Explain array\_flip() function with example.

## strrev()

Function takes a string and returns a reversed copy of it.

**Example:**

```

<?php
echo strrev("Hello World!");
>

```

**Output:**

```

?php
echo strrev("Hello World!");
>
Output:
!dlroWolleH

```

## strpos()

Function returns the position of the first occurrence of a substring in a string.

**Example:**

```

mixed strpos(string $str, mixed $find[, int $offset = 0])

```

The function returns the numeric position of the first occurrence of 'find' in the 'str' string. Mixed means any PHP type. If 'find' is not a string, it is converted to an integer and applied as the ordinal value of a character. 'offset' specifies that search will start this number of characters counted from the beginning of the string.

The function returns the position of where the 'find' exists relative to the beginning of 'str' string (independent of offset). Also note that string positions start at 0 and Returns False if the 'find' was not found.

**Example:**

```

<?php
echo strpos("I am a PHP programmer", "am");
echo "<br>";
echo strpos("I am a PHP programmer", "am", 5);
>

```

**Output:**

```

6

```

Above program the position (index) of character 'I' is 0, hence it displays position of 'am' is 2. In second case, searching starts from 5<sup>th</sup> character, hence 'am' is at position 16.

## strrpos()

Syntax of this function is same as strpos() function. This function finds the last occurrence of a substring in the main string.

**Example:**

```

<?php
echo strrpos("I am a PHP programmer", "am");
echo "<br>";
echo strrpos("I am a PHP programmer", "am", 5);
?>

```

**Output:**

```

16
16

```

# 3...

## Apply Object Oriented Concepts in PHP

### Chapter Outcomes...

- Write constructor and destructor functions for the given problem in PHP.
- Implement inheritance to extend the given base class.
- Use overloading/overriding to solve the given problem.
- Clone the given object.

### Learning Objectives...

- To understand Basic OOP Concepts in PHP
- To learn Classes, Objects, Constructor and Destructor with Example
- To study Concept of Inheritance, Method or Function Overloading and Function Overriding
- To understand Cloning Object, Introspection and Serialization in PHP

### 3.0 INTRODUCTION

- Object Oriented is an approach to software development that models application around real world objects such as employees, cars, bank accounts, etc. A class defines the properties and methods of a real world object. An object is an occurrence of a class.
- PHP is an Object Oriented Programming Language (OOPL). Object-Oriented Programming (OOP), is a type of programming language principle added to PHP, that helps in building complex, reusable web applications.
- OOP, refers to the method of programming that invokes the use of classes to organize the data and structure of an application.
- In OOP, we take a whole different approach. We model our problems and processes using objects. Therefore, an object-oriented application consists of related objects that collaborate with each other to solve the problem.
- In object oriented programming, everything will be around the objects and class. By using OOP in PHP we can create modular web application. By using OOP in PHP we can perform any activity in the object model structure.
- Every object-oriented language seems to have a different set of terms for the same old concepts. This chapter describes the terms that PHP uses, but be warned that in other languages these terms may have other meanings.

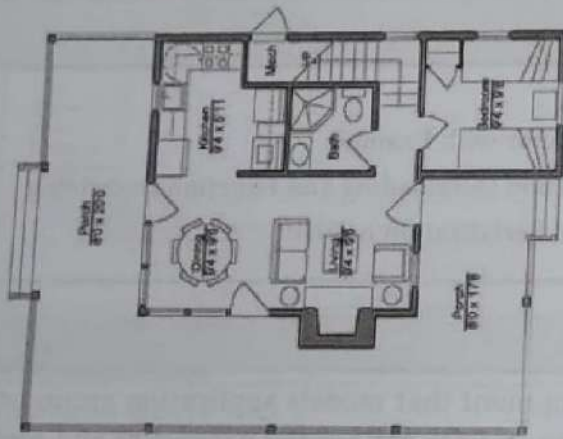
#### Concept of Class:

- The class is a collection of data members and functions. Class contains the actual code that defines the properties and methods of objects. A class defines how an object will behave.

- A class itself is made up of parts called properties and methods.
  1. The properties are different types of data objects like numbers, strings, nulls, and Booleans. Generally, the data are stored as abstract data types known as variables, constants, and arrays.
  2. Methods, on the other hand, are functions that operate on the data.
- **For example:** A bike has a colour, a weight, a manufacturer, and a petrol tank etc. Those are its characteristics. A bike can accelerate, stop, signal for a turn, and sound the horn. Those are its behaviours. Those characteristics and behaviours are common to all bikes. OOP enables us to establish those characteristics through the use of a construct known as a 'Class'. A class describes the characteristics and behaviour of all the members of a set. In OOP, characteristics of a class are known as its 'properties'. Properties have a name and a value.

### Concept of Object:

- An object is an instance or occurrence of a class. Anything in the world is an object like laptop, books, car etc. even object has two thing properties and behaviors.
- The data associated with an object are called its properties. The functions associated with an object are called its methods. We can group similar objects with the same characteristics and behaviors in into one class.
- For example, a class is the blueprint for a house. A blueprint defines characteristics of the house on paper. From the blueprint, we can build as many houses as we want. We say an object is an instance of the class, or a house is an instance of the blueprint.



(a) Blueprint (Class)



(b) House (Objects)

Fig. 3.1: PHP OOP Objects vs. Classes

## 3.1 CREATING CLASSES AND OBJECTS

- Class in PHP is a programmer/user-defined data type. A class as a template for making many instances of the same kind (or class) of object. A class is a template for objects, and an object is an instance of a class.

### 3.1.1 Creating Classes

- A class is defined by using the 'class' keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods goes inside the braces.
- The general **syntax** for defining a new class in PHP is as follows:

```
class classname[extends baseclass]
{
    [ var$property [ = value ]; ... ]
    [ functionfunctionname(args)
    {
        // code...
    }
    ]...
}
```

- The classes are defined with class keyword, followed by the name of the class that we want to define. A set of braces enclosing any number of variable declarations and function definitions. Both variable declaration and function definition are optional. We can also create empty class in PHP.
- Variable declarations start with the special form var, which is followed by a conventional \$variable\_name; they may also have an initial assignment to a constant value.
- Function definitions look much like standalone PHP functions but are local to the class and will access the variable (object data) declared inside the class.

**Example:** For class creation in PHP,

```
<?php
class Student
{
// property declaration
var $roll_no= 1; // 1 is the default value
function showRollNo() // method definition
{
    echo $this->roll_no;
}
}
?>
```

### 3.1.2 Creating Object

- To create an object of a given class, use the 'new' keyword.

**Syntax:** \$object = new classname;

**For example:** \$s1 = new Student;

Here, s1 is the object of class Student.

#### Accessing Properties and Methods:

- Once, we have an object, we can use the -> (object operator) to access properties and methods of the object.

**Syntax:**

```
$object->property_name;
$object->method_name([arg, ... ]);
```

**Example:** For accessing properties and methods of an object.

```
<?php
class Student
{
var $roll_no;
var $name;
function display()
{
    echo "Roll No: " . $this -> roll_no . "<br>";
    echo "Name: " . $this -> name;
}
}
$s1 = new Student;
$s1 -> roll_no = 10; // Properties and methods are public by default
$s1 -> name = "Amar";
$s1 -> display();
?>
```

**Output:**

```
Roll No: 10
Name: Amar
```

- `$this` is a reference to the calling object. `$this` is available inside any class method when that method is called from within an object context.
- Within class methods non-static properties may be accessed by using `->`. Static properties are accessed by using the `::` (Double Colon).
- Static methods and properties in PHP can directly accessible without creating object of class. PHP class will be static class if all methods and properties of the class are static.

### Concept of Visibility:

- The visibility of class members (i.e. properties and methods), relates to how that member may be manipulated within, or from outside the class.
- There are three different levels of visibility that a member variable or method can have i.e., public, private and protected.
  1. **Public Members:** Any property or method which is not explicitly declared as private or protected is a public method. We can access a public method from inside or outside the class.
  2. **Private Members:** Properties or methods declared as private are not allowed to be called from outside the class. However any method inside the same class can access them without a problem.
  3. **Protected Members:** Properties or methods declared as protected are available to class itself and to classes that inherited from it.
- Public is the default visibility level.

## 3.2 CONSTRUCTOR AND DESTRUCTOR

- Constructor refers to a special type of function which will be called automatically whenever there is an object formation/creation from a class. PHP provides a special function called `__construct()` to define a constructor.
- Destructor refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope. We can define a destructor function using function `__destruct()`.

### 3.2.1 Constructor

- When we create a new object, it is useful to initialize its properties. PHP provides us, with a special method to help initialize object's properties called constructor.
- Constructors initialize objects at the time of its creation. To add a constructor to a class, we simply add a special method with the name `__construct()` or use same name as that of class for constructor.
- Whenever, we create a new object, PHP searches for this method and calls it automatically.

**Example:** For constructor.

```
<?php
class Student
{
    public $roll_no;           // var is not needed if visibility is used
    public $name;
    function __construct()    // or function student()
    {
        $this-> roll_no = 10;
        $this-> name = "Amar";
    }
    public function display()
    {
        echo "Roll No: " . $this->roll_no . "<br>";
        echo "Name: " . $this->name;
    }
}

$s1 = new Student;
$s1-> display();

?>
```

**Output:**

Roll No: 10

Name: Amar

- PHP have not yet supported constructor overloading. Fortunately, we can achieve the same constructor overloading effect by using several PHP functions.

**Example:**

```
<?php
class BankAccount
{
    private $accountNumber;
    private $totalBalance;
    public function __construct()
    {
        $args = func_get_args();
        $num = func_num_args();
        if(method_exists($this,$f = 'init_' . $num))
        {
            call_user_func_array(array($this,$f),$args);
        }
    }
    public function init_1($accountNo)
    {
        $this->accountNumber = $accountNo;
    }
    public function init_2($accountNo,$initialAmount)
    {
        $this->accountNumber = $accountNo;
        $this->totalBalance = $initialAmount;
    }
    //... other methods in the below section
}
$a1 = new BankAccount('121412324', 20000);
var_dump($a1);
$a2 = new BankAccount('232321242');
var_dump($a2);
?>
```

**Output:**

```
object(BankAccount)[1]
  private 'accountNumber' => string '121412324' (length=9)
  private 'totalBalance' => int 20000
object(BankAccount)[2]
  private 'accountNumber' => string '232321242' (length=9)
  private 'totalBalance' => null
```

**How the Constructor Works?**

- First, we get constructor's arguments using the `func_get_args()` function and also get the number of arguments using the `func_num_args()` function.
- Second, we check if the `init_1()` and `init_2()` method exists based on the number of constructor's arguments using the `method_exists()` function. If the corresponding method exists, we call it with an array of arguments using the `call_user_func_array()` function.



### 3.2.2 Destructor

- PHP 5 introduces a destructor concept similar to that of C++ language. The destructor method will be called as soon as there are no other references to a particular object.
- PHP destructor allows us to clean up (destroy) resources before PHP releases the object from the memory. For example, we may create a file handle in the constructor and we close it in the destructor.
- To add a destructor to a class, we just simply add a special method called `__destruct()` function.
- There are following some important points regarding to the destructor:
  1. Unlike a constructor, a destructor cannot accept any argument.
  2. Object's destructor is called before the object is deleted. It happens when there is no reference to the object or when the execution of the script is stopped by the `exit()` function.

**Example:** For destructor.

```
<?php
class Test
{
    function __construct()
    {
        print "Constructor called...<br>";
    }
    function __destruct()
    {
        print "Destructor called...";
    }
}
$obj = new Test();
?>
```

**Output:**

```
Constructor called...
Destructor called...
```

### 3.3 INHERITANCE

- PHP supports inheritance, a well-established programming concept. In inheritance, the new class can inherit the properties and methods from the old class.
- The old class is the base class, also called as parent class or super class, and the new class is the derived class also called as child class or sub class.
- The derived class has its own variables and methods plus variables and methods from the base class. The 'extends' keyword is used for the inheritance.

**Example:** For inheritance.

```
class Person
{
    var $name, $age;
}
class Employee extends Person
{
    var $salary, $designation;
}
```

- In the above program, Person is the base class. Employee is the derived class. The employee class contains the \$salary \$designation, as well as the \$name and \$age properties inherited from the person class.

- An extended class is always dependent on the single base class, multiple inheritance is not supported. We can create object of Employee class as follows:  
`$obj_emp = new Employee;`
- Then, `obj_emp -> name`, `obj_emp -> age`, `obj_emp -> salary` // is allowed.
- Create object of person class.  
`$obj_per = new Person;`  
`$obj_per -> name`, `obj_per -> age` // is allowed.  
`$obj_per -> salary` // is not allowed.

**Example:**

```
<?php
class Cat
{
    public $weight;
    public $maxspeed;
    function eat()
    {
        echo "Eating <br>";
    }
    function sleep()
    {
        echo "Sleeping <br>";
    }
}
class Lion extends Cat
{
    public $maneLength;
    function roar()
    {
        echo "Roarrrrrr<br>";
    }
}
$objLion = new Lion();
$objLion->weight = 200;
$objLion->maneLength = 36;
$objLion->eat();
$objLion->roar();
$objLion->sleep();
?>
```

**Output:**

```
Eating
Roarrrrrr
Sleeping
```

**3.3.1 Method or Function Overloading**

- If the derived class is having the same method name as the base class then the method in the derived class takes precedence over or overrides the base class method.
- Function overloading or method overloading is the ability to create multiple functions of the same name with different implementations depending on the types of their arguments.

**Example:** For method overloading.

```
<?php
class Base
{
    function show( )
    {
        echo "Display base <br>";
    }
}
class Derived extends Base
{
    function show( )
    {
        echo "Display Derived";
    }
}
$obj = new Derived();
$obj -> show( );
?>
```

**Output:**

Display Derived

- In above program, both the classes define the same method show(), but when we inherit, the derived version of show() overrides the base version of show(). When we call show() method, it always calls the derived show() method.
- To call base class show() method we can use parent::method(). The scope resolution operator (::) is used to refer functions and variables in base class or to refer functions in classes that have not yet any instances.
- In the above program, the derived version of show() method can be modified as:

```
function show()
{
    parent:: show(); // Display base
    echo "Display Derived";
}
```

- PHP does not provide for automatic chain of constructor. Only constructor in the derived class is automatically called.
- Parent constructors are not called implicitly if the child class defines a constructor. In order to run a parent constructor, a call to parent::\_\_construct() within the child constructor is required. If the parent class has defined a method as final, that method may not be overridden.

### 3.3.2 Method or Function Overriding

- In function overriding, both parent and child classes should have same function name with and number of arguments. It is used to replace parent method in child class.
- The purpose of overriding is to change the behavior of parent class method. The two methods with the same name and same parameter is called overriding.
- In short, the two methods in PHP programming with the same name and same parameter is called overriding.

**Example:** For overloading.

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

```
// This is parent class
classP {
    // Function geeks of parent class
    function geeks() {
        echo"Parent";
    }
}
// This is child class
classC extendsP
{
    // Overriding geeks method
    function geeks()
    {
        echo"\nChild";
    }
}
// Reference type of parent
$P= newP;
// Reference type of child
$c= newC;
// print Parent
$P->geeks();
// Print child
$c->geeks();
?>
```

**Output:**

Parent  
Child

### 3.3.3 Cloning Object

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

**Syntax:** `$copy_object_name = clone $object_to_be_copied`

**Example:**

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

```

    // Creating clone or copy of object
    $copy= clone$obj;
    // Set values of $obj object
    $obj->data1 = "PHP";
    $obj->data2 = "with";
    $obj->data3 = "Web";
    // Set values of copied object
    $copy->data1 = "Computer";
    $copy->data2 = "IT ";
    $copy->data3 = "Portal";
    // Print values of $obj object
    echo "$obj->data1$obj->data2$obj->data3\n";
    // Print values of $copy object
    echo "$copy->data1$copy->data2$copy->data3\n";
?>

```

**Output:**

```

PHP with Web
Computer IT Portal

```

**3.4 INTROSPECTION**

- Introspection is the ability of a program to examine an object's characteristics, such as its name, parent class (if any), properties and methods.
- Introspection allows us to:
  1. Obtain the name of the class to which an object belongs as well as its member properties and methods.
  2. Write generic debuggers, serializers, profilers etc.
- Introspection in PHP offers the useful ability to examine classes, interfaces, properties, and methods. With introspection, we can write code that operates on any class or object.

**3.4.1 Examining Classes**

- To examining classes the introspective functions provided by PHP are `class_exists()`, `get_class_methods()`, `get_class_vars()` etc.

**1. class\_exists():**

- This function is used to determine whether a class exists. It takes a string and return a Boolean value.

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

- This function returns TRUE if classname is a defined class, FALSE otherwise.

For example,

```

<?php
    if (class_exists('MyClass'))
    {
        $myclass = new MyClass();
    }
?>

```

- Alternately, we can use the `get_declared_classes()` function.

`$classes = get_declared_classes();`

- This function returns an array of defined classes.

**2. get\_class\_methods():**

- This function returns an array of class methods name.

**Syntax:** `$methods = get_class_methods(classname);`

For example,

```
<?php
class myclass
{
    function myclass(){}
    function myfunc1(){}
}
$my_class = new myclass();
$class_methods = get_class_methods('myclass');
print_r($class_methods); // Array ( [0] =>myclass [1] => myfunc1 )
?>
```

### 3. get\_class\_vars():

- This function returns an array of default properties of the class. It returns an associative array with property's name value pairs.

**Syntax:** `$properties = get_class_vars(classname);`

For example,

```
<?php
class myclass
{
    var $v1; // $v1 is not set, NULL value is set by default
    var $v2 = 100;
}
$my_class = new myclass();
$class_vars = get_class_vars('myclass');
var_dump($class_vars); // array(size=2)'v1' => null'v2' =>int 100
?>
```

### 4. get\_parent\_class():

- Accept either an object or class name as parameter. It returns the name of the parent class, or false if there is no parent class.

**Syntax:** `$superclass = get_parent_class (classname);`

## 3.4.2 Examining an Object

- To examining an object the introspection uses following functions:

#### 1. is\_object(): Return True if passed parameter is an object.

**Syntax:** `$yes_no = is_object(var);`

#### 2. get\_class(): Returns the name of the class of an object.

**Syntax:** `$class_name = get_class($object);`

Returns False if object name passed as the parameter is not an object.

#### 3. get\_object\_vars(): Gets the properties of the given object.

**Syntax:** `$array = get_object_vars($object);`

#### 4. method\_exists(): Checks if the class method exists.

**Syntax:** `$yes_no = method_exists($object, $method_name);`

## 3.5 SERIALIZATION

- Serializing an object means converting it to a byte stream representation that can be stored into a file. This is useful for persistent data. For example, PHP sessions automatically save and restore objects.

- We can use two functions `serialize()` and `unserialize()` to implement our own form of persistent objects.

For example,

```
$encode = serialize (somedata);
$somedata = unserialize (encode);
```

- `serialize()` method returns a string containing a byte-stream representation of any value that can be stored in PHP. `unserialize()` method can use this string to recreate the original variable values.

**Syntax:** `string serialize(mixed $value)`

- While saving an object, all the variables of the object will be saved, but the functions will not be saved. After serialization of an object the byte version can be stored in a file.
- To `unserialize()` an object in another PHP file, the class must be defined in that file. This can be done by including file where the class has been defined.

**Syntax:** `mixed unserialize(string $string)`

- The following program shows how object is serialized and stored in a text file, then the object is unserialized.

```
<?php
class Student
{
    var $age = 10;
    function show_Age()
    {
        echo $this ->age;
    }
}
$stud = new Student;
$stud->show_Age(); // Outputs 10
$sri_obj = serialize ($stud); // $stud is serialized, converted to string
$fp = fopen ("a.txt", "w");
fwrite ($fp, $sri_obj); // storing serialized obj into the file
fclose ($fp);
$us_obj = unserialize($sri_obj); // new object $us_obj created
$us_obj->show_Age(); // Outputs 10
?>
```

- PHP has two hooks for objects during the serialization and unserialization process: `__sleep()` and `__wakeup()`. These methods are used to notify objects that they are being serialized or unserialized.

**`__sleep()` and `__wakeup()` Functions:**

- These functions are used to notify objects that they are being serialized or unserialised. `serialize()` checks if our class has a function with the magic name `__sleep()`. If so, that function is executed prior (i.e. just before) to any serialization.
- It can clean up the object and is supposed to return an array with the names of all variables of that object that should be serialized. If the method doesn't return anything then Null is serialized. The `__sleep()` function is useful if we have very large objects which do not need to be saved completely.
- `unserialize()` checks for the presence of a function with the magic name `__wakeup()`. If present, this function can reconstruct any resources that the object may have.

**Example:**

```
<?php
class Customer
{
```

```
private $name;
private $credit_card_no;
public function __construct($name, $credit_card_no)
{
    $this->name = $name;
    $this->credit_card_no = $credit_card_no;
}
public function __sleep()
{
    echo "Sleep calling <br>";
    return array('name', 'credit_card_no');
}

public function __wakeup()
{
    echo "Wakeup calling <br>";
}
}
$c = new Customer("Amar", 1234567890);
$data = serialize($c);
echo $data . "<br>";
print_r(unserialize($data));
?>
```

**Output:**

Sleep calling

O:8:"Customer":2:{s:14:"Customername";s:4:"Amar";s:24:"Customercredit\_card\_no";i:1234567890;}

Wakeup calling

Customer Object ([name:Customer:private] =>Amar [credit\_card\_no:Customer:private] =>1234567890 )

- In above program we serialize name and credit\_card\_no both. To serialize any one of the property just pass that property name into the array.

**Practice Questions**

1. What is OOP?
2. What is object?
3. Explain class with example.
4. How to create an object? Explain with example.
5. What is inheritance?
6. Describe method overloading with example.
7. Explain sleep and wakeup functions?
8. What serialization in PHP?



# 4...

## Creating and Validating Forms

### Chapter Outcomes...

- Use the relevant form controls to get user's input.
- Design web pages using multiple forms for the given problem.
- Apply the given validation rules on form.
- Set/modify/delete cookies using cookies attributes.
- Manage the given session using session variables.

### Learning Objectives...

- To understand Forms in PHP with Form Controls (Textbox, Radio Button, Checkbox etc.)
- To learn creating a Webpage using GUI Components
- To Working with Multiple Forms in PHP
- To study Form and Web Page Validation in PHP
- To understand Basic Concepts in Cookies with its Types
- To learn Concepts of Session in PHP

### 4.0 INTRODUCTION

- Nowadays, PHP is becoming a standard in the world of web programming with its simplicity, performance, reliability, flexibility and speed. One of the most powerful feature of PHP is the way it handles HTML forms.
- Forms are essential parts in web development. Forms are used to get input from the user and submit it to the web server for processing. Forms are used to communicate between users and the server.
- Fig. 4.1 shows the form handling process.

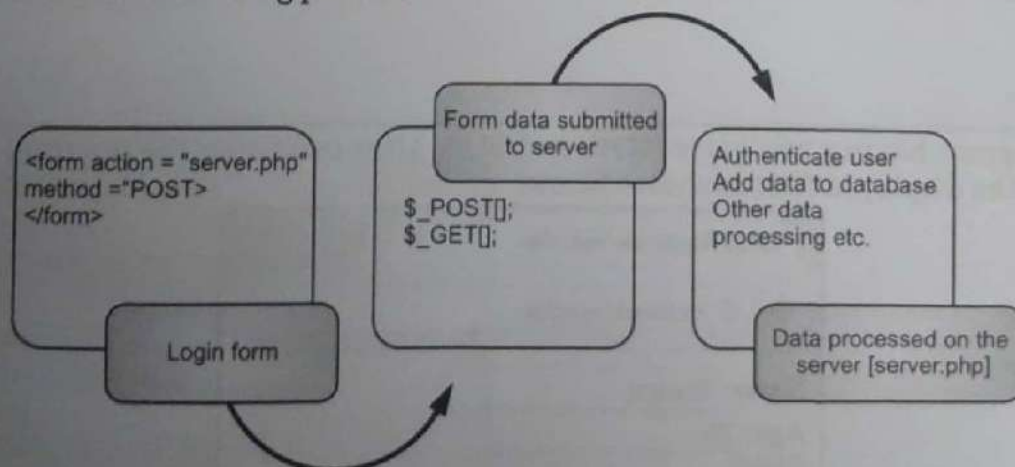


Fig. 4.1: Process of Form Handling  
[4.1]

- A form is an HTML tag that contains Graphical User Interface (GUI) items such as input box, check boxes, radio buttons etc. The basic advantage of using PHP script is it automatically identifies the form element.
- The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

#### 4.1 CREATING A WEBPAGE USING GUI COMPONENTS

- A document that containing blank fields, that the user can fill the data or user can select the data. It is known as Form.
- Generally, the data will store in the database. We can create and use forms in PHP. To get form data, we need to use PHP superglobals \$\_GET and \$\_POST.
- The form request may be get or post. To retrieve data from get request, we need to use \$\_GET, for post request \$\_POST.

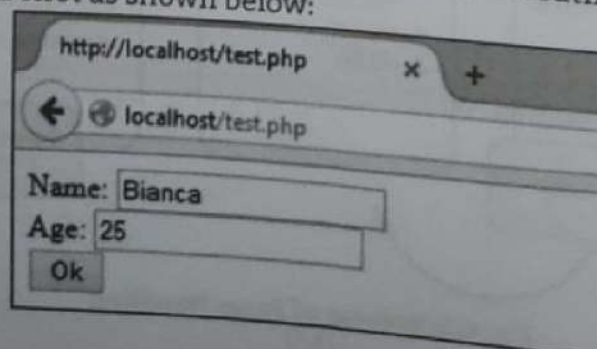
##### 4.1.1 Browser Role GET and POST Methods

- There are two HTTP methods that a client can use to pass form data to the server i.e., GET and POST. The method is specified with the method attribute to the <form> tag.
- 1. GET Method:**
    - The GET method sends the encoded user information appended to the page request (to the URL). The page and the encoded information are separated by the ? character.  
http://www.test.com/index.htm?name1=value1&name2=value2
    - The GET method produces a long string that appears in our server logs, in the browser's Location: box. The GET method is restricted to send up to 1024 characters only.
    - Never use GET method if we have password or other sensitive information to be sent to the server. 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.

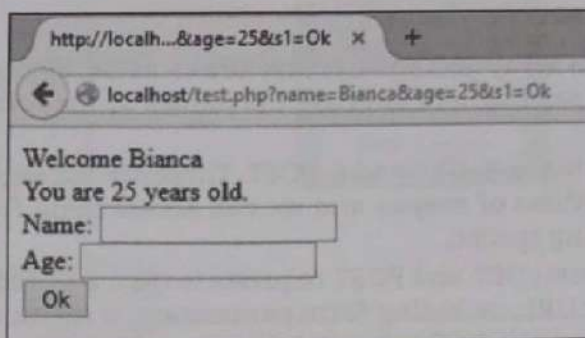
**Example:** For putting the source code in test.php script.

```
<?php
if(isset($_GET["s1"]) )
{
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";
}
?>
<html>
<body>
<form action="<?php $_PHP_SELF?>" method="GET">
    Name: <input type="text" name="name" /><br>
    Age: <input type="text" name="age" /><br>
    <input type="submit" name="s1" value="Ok"/>
</form>
</body>
</html>
```

- The above program having two parts HTML and PHP. After executing the program in the browser, HTML part will be displayed first as shown below:



- Enter name and age and submit the form. Now PHP will process the form and we will get the following output:



- After submitting the form the URL in the address bar is as follows:  
http://localhost/test.php?name=Bianca&age=25&s1=Ok  
It shows all the three form parameter's name/value pairs.
- In the program, the isset() method is used to check whether the "Ok" button is pressed or not. After entering name and age submitting the form will store the entered value of the text field into the \$\_GET array, and that can be displayed.

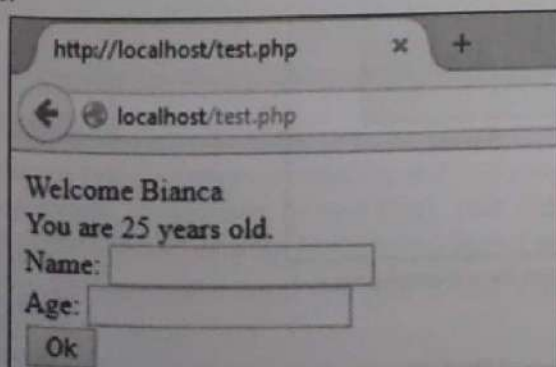
## 2. POST Method:

- The POST method transfers information via HTTP headers, not through the URL. The information is encoded as described in case of GET method and put into a header called QUERY\_STRING.
- The POST method does not have any restriction on data size to be sent. The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP we can make sure that our information is secure. The PHP provides \$\_POST associative array to access all the sent information using POST method.

**Example:** For putting the source code in test.php script.

```
<?php
if(isset($_POST["s1"]) )
{
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
}
?>
<html>
<body>
<form action="<?php $_PHP_SELF?>" method="POST">
    Name: <input type="text" name="name" /><br>
    Age: <input type="text" name="age" /><br>
    <input type="submit" name="s1" value="Ok"/>
</form>
</body>
</html>
```

**Output:**



- The output of the above program is same, except URL, the URL in the address bar will not get changed after submitting the form.

#### GET vs. POST Methods:

- Both GET and POST create an array (example, array( key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Both GET and POST are treated as `$_GET` and `$_POST`. These are superglobals, which means that they are always accessible, regardless of scope - and we can access them from any function, class or file without having to do anything special.
- The biggest difference between GET and POST requests is that, GET requests are idempotent i.e., one 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 does not change regardless of how many times the page is loaded. Because of idempotence, GET requests should be used only for queries such as splitting a word into smaller chunks or multiplying numbers, where the response page is never going to change.
- POST requests are not idempotent. This means that they cannot be cached, and the server is recontacted every time the page is displayed. We have probably seen the web browser prompt us with "Repost form data?" before displaying or reloading certain pages. This makes POST requests the appropriate choice for queries whose response pages may change over time—for example, displaying the contents of a shopping cart.

#### When to use GET?

- Information sent with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.

**Note:** GET should NEVER be used for sending passwords or other sensitive information.

#### When to use POST?

- Information sent with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

### 4.1.2 Server Role

- PHP is a server-side scripting language. Using PHP we can create dynamic web sites. That means its processing happens in the server by consuming server's resources and sends only the output to the client.
- In a client-side scripting language like JavaScript, processing happens in the client's computer consuming its resources.

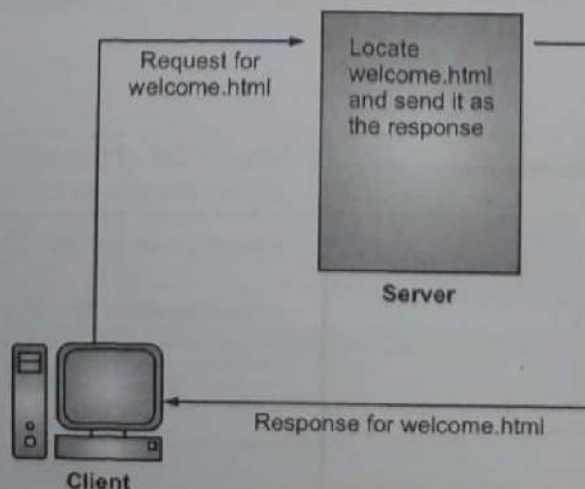


Fig. 4.2: Cycle of a Normal Web Request

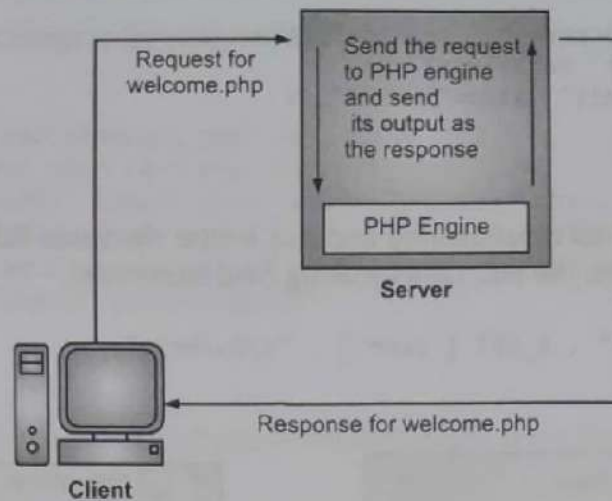


Fig. 4.3: Cycle of a Web Request that involves PHP

**Example:****welcome.php**

```

<html>
<head>
<title>Welcome to Our Web Site</title>
</head>
<body>
<h1>
<?php
    if (date('G') < 12)
    {
        echo 'Good Morning!';
    }
    else
    {
        echo 'Welcome!';
    }
?>
</h1>
<p>Rest goes here...</p>
</body>
</html>
  
```

**Output:****4.2 FORM CONTROLS**

- There are different types of form controls that we can use to collect data using form namely, Textbox, Checkbox, Radio button, and so on.

**4.2.1 Textbox**

- A text input field allows the user to enter a single line of text.
- In other words, a text box (input box), text field (input field) or text entry box is a graphical control element intended to enable the user to input text information to be used by the program.

**Example:** For welcome.html file and it has a text field and submit button.

```

<html>
<body>
  
```

```

<form action="welcome.php" method="get">
  <input type="text" name="user" />
  <input type="submit" value="SUBMIT" />
</form>
</body>
</html>

```

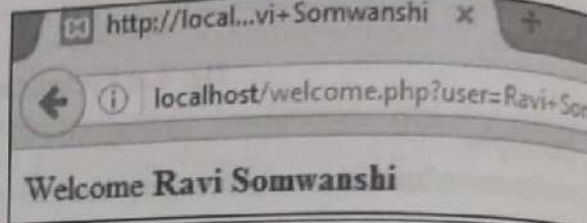
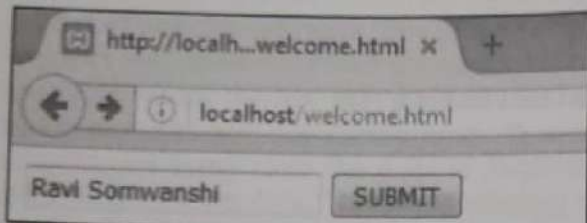
Name the following script as welcome.php and put it into the same folder as above welcome.html file. It accepts the value from the text field by using field name user.

```

<?php
  echo "Welcome <b>" . $_GET ['user'] . "</b><br/>";
?>

```

**Output:**



### 4.2.2 Textarea

- A textarea field is similar to a text input field, but it allows the user to enter multiple lines of text.
- Unlike most other controls, an initial value is placed between the <textarea> ... </textarea> tags, rather than in a value attribute.

**Example:** Name the following script as textarea.html. It has a text area and a submit button.

```

<html>
  <body>
    <form action="textarea1.php" method="get">
      <textarea name="address" rows="5" cols="40"></textarea>
      <input type="submit" value="SUBMIT" />
    </form>
  </body>
</html>

```

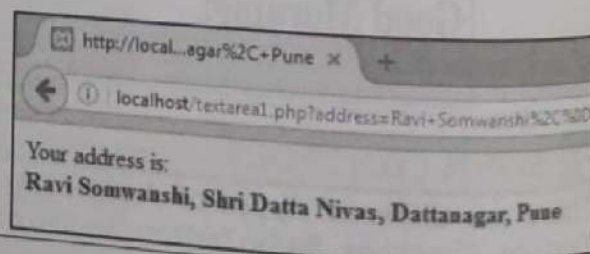
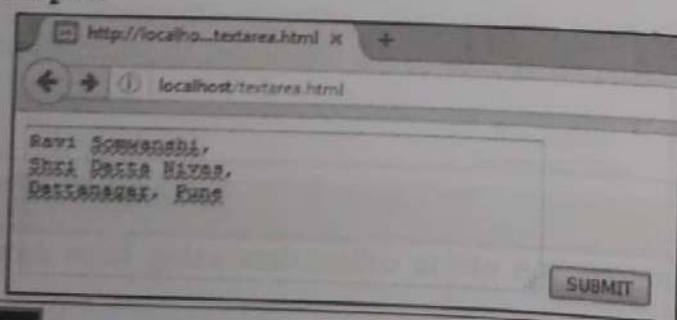
Name the following script as textarea1.php. It reads the value from the textarea from the form above.

```

<?php
  echo "Your address is: <br/><b>" . $_GET ['address'] . "</b>";
?>

```

**Output:**



### 4.2.3 Radio Button

- The radio buttons are for single choice from multiple options. All radio buttons in the group have the same name attribute.
- Only one button can be selected per group. As with checkboxes, use the value attribute to store the value that is sent to the server if the button is selected.
- The value attribute is mandatory for checkboxes and radio buttons, and optional for other field types.

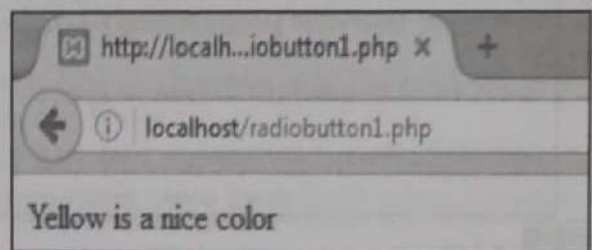
**Example:** The following script is for radiobutton.html. It has a group of radio buttons.

```
<html>
  <body>
    <form action="radiobutton1.php" method="post">
      <b>Please select your favorite color</b><br>
      <input type="radio" name="color" value="White"> White
      <input type="radio" name="color" value="Yellow"> Yellow
      <input type="radio" name="color" value="Red"> Red <br>
      <input type="submit" value="SUBMIT">
    </form>
  </body>
</html>
```

The following script is for radiobutton1.php. It reads the data from the form above.

```
<?php
  $color = $_POST['color'];
  if( ( $color != null ) )
  {
    echo $color." is a nice color";
  }
?>
```

**Output:**



#### 4.2.4 Checkbox

- A checkbox field is a simple toggle button. It can be either ON or OFF. The value attribute should contain the value that will be sent to the server when the checkbox is selected. If the checkbox isn't selected, nothing is sent.
- By creating multiple checkbox fields with the same name attribute, we can allow the user to select multiple values for the same field.

**Example:** The following script is for **checkbox.html**. It has several checkboxes.

```
<html>
  <head>
    <title>Multiple Checkboxes</title>
  </head>
  <body>
    <div>
      <form action="checkbox.php" method="post">
        <h4>Select your interested programming language:</h4>
        <input type="checkbox" name="check_list[]" value="C/C++"><label>C/C++</label>
        <input type="checkbox" name="check_list[]" value="Java"><label>Java</label>
        <input type="checkbox" name="check_list[]" value="PHP"><label>PHP</label>
        <input type="checkbox" name="check_list[]" value="Perl"><label>Perl</label>
        <input type="checkbox" name="check_list[]"
          value="Python"><label>Python</label><br><br>
        <input type="submit" name="submit" Value="SUBMIT"/>
      </form>
    </div>
  </body>
</html>
```

**checkbox.php**

```

<?php
if(isset($_POST['submit']))
{
if(!empty($_POST['check_list']))
{
    // Counting number of checked checkboxes.
    $checked_count = count($_POST['check_list']);
    echo "You have selected following ".$checked_count." option(s): <br/>";
    foreach($_POST['check_list'] as $selected)
    {
        echo "<p>".$selected."</p>";
    }
}
else
{
    echo "<b>Please Select Atleast One Option.</b>";
}
}
?>

```

**Output:**
**4.2.5 List**

- The list represents a Windows control to display a list of items to a user. A user can select an item from the list. Users can either select one option from a list or multiple options, depending on the type of list.
- A multi-select list allows the user to select multiple items at once by holding down Ctrl or Command key. To turn a normal list into a multi-select list, add the attribute multiple with a value of "multiple" to the select element.

**Example:** For list**listbox.html**

```

<html>
<body>
<div>
    <form action="listbox.php" method="post">
    <p><select name="Transport[]" multiple="multiple">
    <option>BUS</option>
    <option>CAR</option>
    <option>RAILWAY</option>
    <option>AROPLANE</option>
    </select></p>
    <p><input type="submit" value="SUBMIT" /></p>
    </form>
</div>
</body>
</html>

```

**listbox.php**

```

<?php

```

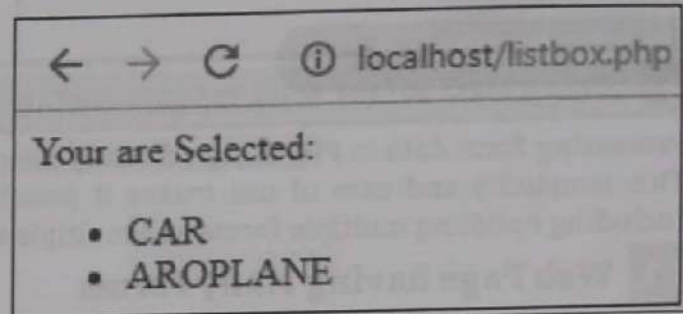
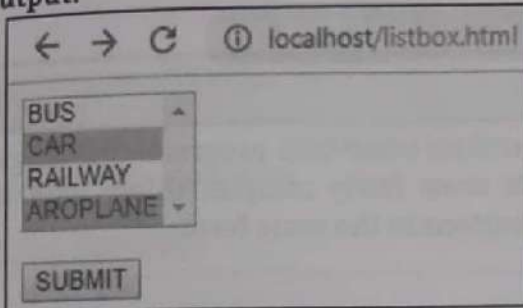


```

if (is_array ( $_POST ['Transport'] ))
{
print "<p>Your are Selected:</p>";
print "<ul>";
foreach ( $_POST ['Transport'] as $value )
{
print "<li>$value</li>\n";
}
print "</ul>";
}
?>

```

Output:



#### 4.2.6 Buttons

- A button is a control, which is an interactive component that enables users to communicate with an application which we click and release to perform some actions.
- The button control represents a standard button that reacts to a Click event. A button can be clicked by using the mouse, ENTER key, or SPACEBAR if the button has focus.
- The <button> element is used to create an HTML button. Any text appearing between the opening and closing tags will appear as text on the button. No action takes place by default when a button is clicked.

**Syntax:** <button type="button|submit|reset">

**Attribute Values:**

**Button:** The button is a clickable button.

**Submit:** The button is a submit button (submits form-data).

**Reset:** The button is a reset button (resets the form-data to its initial values).

**Example:** For button.

```

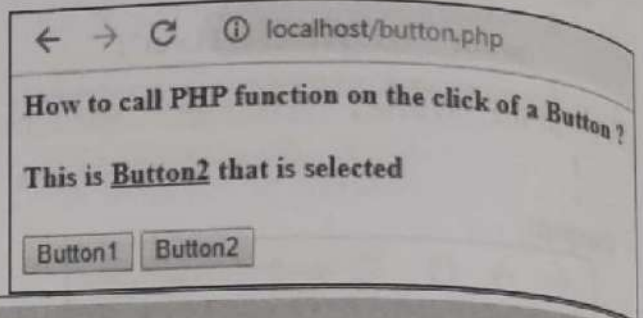
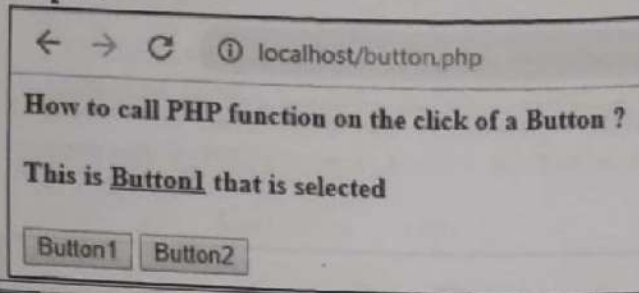
<html>
<head><title> Button Demo </title></head>
<body>
<h4> How to call PHP function on the click of a Button ?</h4>
<?php
    if(array_key_exists('button1', $_POST)) {
        button1();
    }
    else if(array_key_exists('button2', $_POST)) {
        button2();
    }
function button1() {
echo "<h4>This is <u>Button1</u> that is selected</h4>";
}
function button2() {
echo "<h4>This is <u>Button2</u> that is selected</h4>";
}
?>
<form method="post">
<input type="submit" name="button1" class="button" value="Button1" />

```

```

<input type="submit" name="button2" class="button" value="Button2" />
</form>
</body>
</html>

```

**Output:****4.3 WORKING WITH MULTIPLE FORMS**

- Processing form data in PHP is significantly simpler than most other Web programming languages. This simplicity and ease of use makes it possible to do some fairly complex things with forms, including handling multiple forms and multiple submit buttons in the same form.

**4.3.1 Web Page having Many Forms**

- Multiple-page applications work in a "traditional" way. Every change for example, display the data or submit data back to server requests rendering a new page from the server in the browser.
- These applications are large, bigger than single page application because they need to be. Due to the amount of content, these applications have many levels of User Interface (UI).
- Now it is possible to create multiple forms under a single web page to reduce the parameter such as no page reloads, no extra wait time etc.

**Example:**

```

<html>
<body>
    <form name="mailinglist" method="post">
        <fieldset>
            <legend>Form 1:</legend>
            <h4>Email: <input type="text" name="email" /></h4>
            <input type="submit" name="mailing-submit" value="Add our mailing list" />
        </fieldset>
    </form>
    <form name="contactus" method="post">
        <fieldset>
            <legend>Form 2:</legend>
            <h4>Email: <input type="text" name="email" /></h4>
            <h4>Subject:<input type="text" name="subjet" /></h4>
            <h4>Write message here:<textarea name="message"></textarea></h4>
            <input type="submit" name="contact-submit" value="Send Email" />
        </fieldset>
    </form>
    <?php
        if (!empty($_POST['mailing-submit']))
        {
            echo "<h4>Form 1 is Calling</h4>";
        }
        if (!empty($_POST['contact-submit']))
        {
            echo "<h4>Form 2 is Calling</h4>";
        }
    ?>
</body>
</html>

```

## Output:

localhost/multipleforms.php

Form 1:

Email:

Add our mailing list

Form 2:

Email:

Subject:

Write message here:

Send Email

Form 2 is Calling

### 4.3.2 Form having Multiple Submit Buttons

- First let we know, how to use multiple submit buttons in a HTML form and how PHP handle it. Usually a HTML form has only one submit button but there are situations when we might need to use more than one submit buttons and PHP check which button has been pressed and an action to be done according to the button pressed.
- Having multiple submit buttons and handling them through PHP is just a matter of checking the name of the button with the corresponding value of the button using conditional statements.

#### Example:

```

<html>
<head><title> Button Demo </title></head>
<body>
<h4> Form having multiple submit buttons </h4>
<?php
    switch($_REQUEST['btn_submit'])
    {
        case "Button 1":
            echo "<h4>You pressed Button 1</h4>";
            break;
        case "Button 2":
            echo "<h4>You pressed Button 2</h4>";
            break;
        case "Button 3":
            echo "<h4>You pressed Button 3</h4>";
            break;
    }
?>
<form method="post">
<input type="submit" name="btn_submit" class="button" value="Button 1" /><input
type="submit" name="btn_submit" class="button" value="Button 2" />
<input type="submit" name="btn_submit" class="button" value="Button 3" />
</form>
</body>
</html>

```

**Output:**
**4.4 WEB PAGE VALIDATION**

- Validation means check the input submitted by the user. There are two types of validation are available in PHP.

Two types of validations are:

- Client-Side Validation:** Validation is performed on the client machine web browsers.
- Server Side Validation:** After submitted the data, the data has sent to a server and perform validation checks in server machine.

- Some of Validation Rules for fields:

Sr. No	Field	Validation Rules
1.	Name	Should required letters and white-spaces.
2.	Email	Should required @ and.
3.	Website	Should required a valid URL.
4.	Radio	Must be selectable at least once.
5.	Check Box	Must be checkable at least once.
6.	Drop Down menu	Must be selectable at least once.

**Example:** For form validation.

```

<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
    // define variables and set to empty values
    $nameErr = $emailErr = $genderErr = $websiteErr = "";
    $name = $email = $gender = $comment = $website = "";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if (empty($_POST["name"])) {
            $nameErr = "Name is required";
        }else {
            $name = test_input($_POST["name"]);
        }
        if (empty($_POST["email"])) {
            $emailErr = "Email is required";
        }else {
            $email = test_input($_POST["email"]);
            // check if e-mail address is well-formed
            if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
                $emailErr = "Invalid email format";
            }
        }
        if (empty($_POST["website"])) {
            $website = "";

```

```

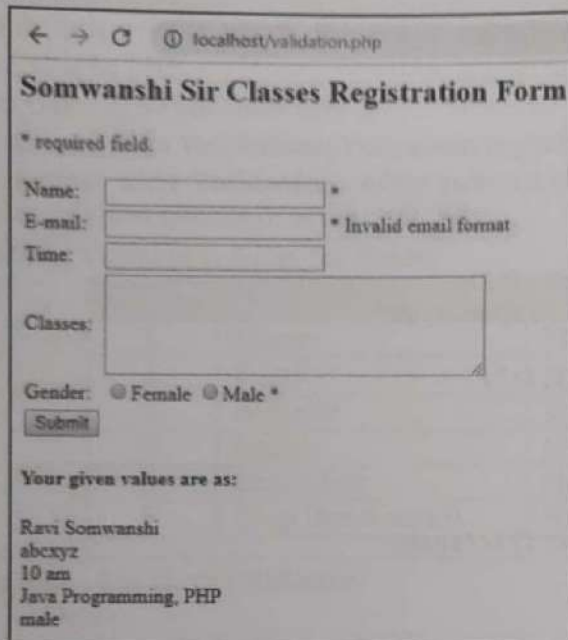
    }else {
    $website = test_input($_POST["website"]);
    }
    if (empty($_POST["comment"])) {
    $comment = "";
    }else {
    $comment = test_input($_POST["comment"]);
    }
    if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
    }else {
    $gender = test_input($_POST["gender"]);
    }
    }
    functiontest_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
    }
?>
<h2>Somwanshi Sir Classes Registration Form</h2>
<p><span class = "error">* required field.</span></p>
<form method = "post" action = "<?php
echohtmlspecialchars($_SERVER["PHP_SELF"]);?>">
<table>
<tr>
<td>Name:</td>
<td><input type = "text" name = "name">
<span class = "error">* <?php echo $nameErr;?></span>
</td>
</tr>
<tr>
<td>E-mail: </td>
<td><input type = "text" name = "email">
<span class = "error">* <?php echo $emailErr;?></span>
</td>
</tr>
<tr>
<td>Time:</td>
<td><input type = "text" name = "website">
<span class = "error"><?php echo $websiteErr;?></span>
</td>
</tr>
<tr>
<td>Classes:</td>
<td><textarea name = "comment" rows = "5" cols = "40"></textarea></td>
</tr>
<tr>
<td>Gender:</td>
<td>
<input type = "radio" name = "gender" value = "female">Female
<input type = "radio" name = "gender" value = "male">Male
<span class = "error">* <?php echo $genderErr;?></span>
</td>
</tr>
<tr>
<td>
<input type = "submit" name = "submit" value = "Submit">
</td>

```

```

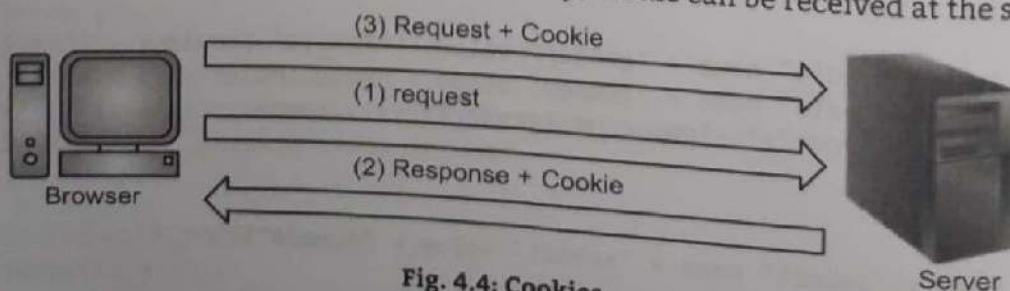
</table>
</form>
<?php
    echo "<h4>Your given values are as:</h4>";
    echo $name."<br>";
    echo $email."<br>";
    echo $website."<br>";
    echo $comment."<br>";
    echo $gender."<br>";
?>
</body>
</html>
    
```

**Output:**



**4.5 COOKIES**

- A cookie is a small piece of data in the form of a name-value pair that is sent by a Web server and stored by the browser on the client machine.
- This information is used by the application running on the server side to customize the web page according to the past history of transactions from that client machine.
- A server can send one or more cookies to a browser in the headers of a response. Some of the cookie's fields indicate the pages for which the browser should send the cookie as part of the request.
- Servers can store any data they like in the value field of the cookie (within limits), such as a unique code identifying the user, preferences, etc.
- PHP cookie is a small piece of information which is stored at client browser in text format. It is used to recognize the user. Cookies are also known as web cookies, HTTP cookies or browser cookies.
- Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



**Fig. 4.4: Cookies**

- In short, cookie can be created, sent and received at server end. Use the `setcookie()` function to send a cookie to the browser. Like `header()`, `setcookie()` must be called before any HTML is printed to the client browser because cookies are set in the HTTP headers, which must be sent before any HTML.
- The `setcookie()` function can be used to delete a cookie. For deleting a cookie, the `setcookie()` function is called by passing the cookie name and other arguments or empty strings but however this time, the expiration date is required to be set in the past.

### 4.5.1 Types of Cookies

- The types of cookies are explained below:

#### 1. Session Cookies:

- Session cookies also called a transient cookie, a cookie that is erased when we close the Web browser.
- The session cookie is stored in temporary memory and is not retained after the browser is closed. Session cookies do not collect information from the computer.
- Session cookies, are temporary means they are stored temporarily in memory and are automatically removed when the browser closes or the session ends.
- Session cookies expire at the end of the session. This means, when we close our browser window, the session cookie is deleted. This website only uses session cookies.

#### 2. Persistent Cookies:

- Persistent cookies do not expire at the end of the session. Persistent cookies also called a permanent cookie, or a stored cookie. A cookie that is stored on the hard drive until it expires (persistent cookies are set with expiration dates) or until we delete the cookie.
- Persistent cookies are used to collect identifying information about the user, such as Web surfing behavior or user preferences for a specific Web site.

### 4.5.2 Use of Cookies

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, we can both create and retrieve cookie values.
- HTTP is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer. The path where the cookies are stored depends on the browser. Internet Explorer usually stores them in Temporal Internet Files folder.
- Personalizing the user experience – this is achieved by allowing users to select their preferences. The page requested that follow are personalized based on the set preferences in the cookies.
- Cookies are used for tracking the pages visited by a user.

#### `isset()` Function:

- To read data from a cookie, we first have to check if the cookie actually exists. This is achieved through the `isset()` function.
- The `isset()` function is used to check for the existence of a variable, in this case, a cookie variable through the use of the `$_COOKIE` associative array which stores an array of existing cookies.

#### Syntax:

```
isset($_COOKIE['nameOfCookie']);
```

- If the cookie specified in the `isset()` function exists, then the function will return true, otherwise it will return false.

#### Example:

```
<?php
if (isset($_COOKIE['cookie1']))
{
$cookie1 = $_COOKIE['cookie1'];
}
?>
```

- In the above example, an if statement checks for the existence of a cookie named cookie1. If it exists, then its value will be passed to the variable \$cookie1. If it does not, then it will remain empty. The isset() function checks for this.

### 4.5.3 Attribute of Cookies

- The following **syntax** shows attributes of cookies:  
setcookie(name, value, expire, path, domain, secure, httponly);
- Only the name parameter is required. All other parameters are optional. Here, is the detail of all the arguments in above syntax:
  - **Name:** Name of the cookie, stored in an environment variable called \$\_COOKIE. This variable is used while accessing cookies.
  - **Value:** Value of the named variable.
  - **Expiry:** A UNIX timestamp denoting the time at which the cookie will expire.
  - **Path:** This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
  - **Domain:** The browser will return the cookie only for URLs within this domain. The default is the server hostname.
  - **Secure:** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

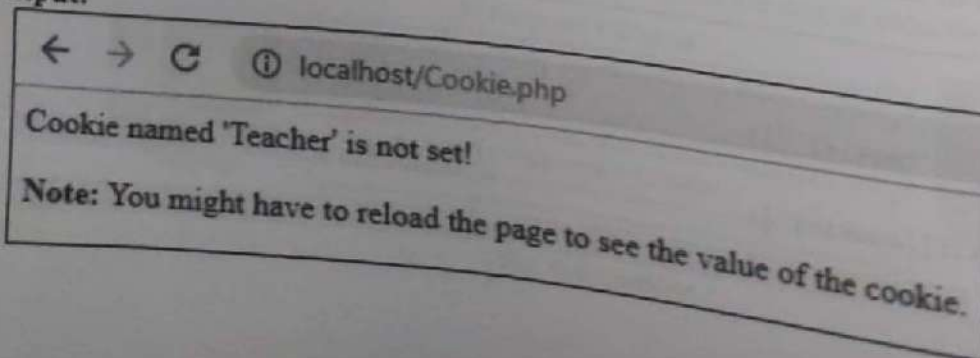
### 4.5.4 Create Cookies

- A cookie is created with the setcookie() function.

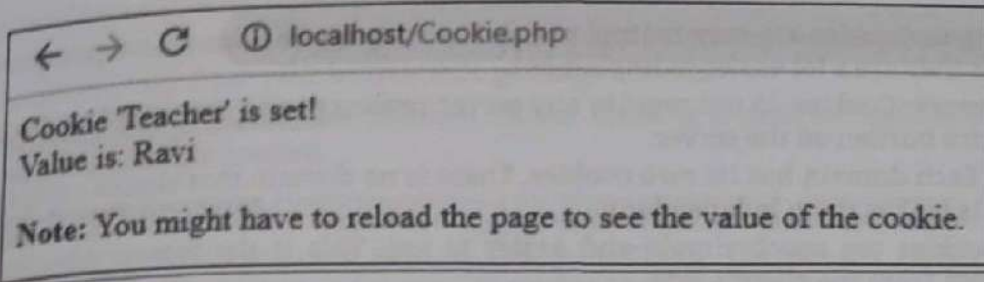
**Example:** For creating cookies.

```
<!DOCTYPE html>
<?php
    $cookie_name = "Teacher";
    $cookie_value = "Ravi";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 =
    1 day.
?>
<html>
<body>
<?php
    if(!isset($_COOKIE[$cookie_name]))
    {
        echo "Cookie named '" . $cookie_name . "' is not set!";
    }
    else
    {
        echo "Cookie '" . $cookie_name . "' is set!<br>";
        echo "Value is: " . $_COOKIE[$cookie_name];
    }
?>
<p><strong>Note:</strong> You might have to reload the page to see the value of the
cookie.</p>
</body>
</html>
```

**Output:**







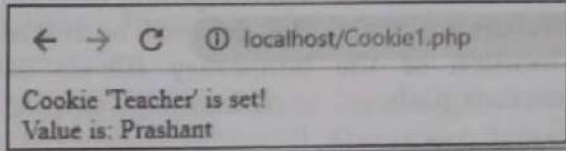
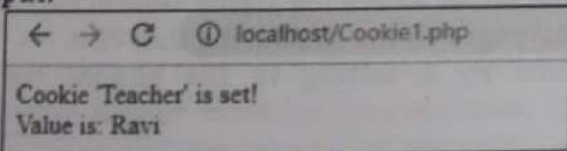
### 4.5.5 Modify Cookies Value

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

**Example:** For modifying cookies.

```
<?php
    $cookie_name = "Teacher";
    $cookie_value = "Prashant";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>
<?php
    if(!isset($_COOKIE[$cookie_name]))
    {
        echo "Cookie named '" . $cookie_name . "' is not set!";
    }
    else
    {
        echo "Cookie '" . $cookie_name . "' is set!<br>";
        echo "Value is: " . $_COOKIE[$cookie_name];
    }
?>
</body>
</html>
```

**Output:**



### 4.5.6 Delete Cookies

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

**Example:** For delete cookies.

```
<!DOCTYPE html>
<?php
    // set the expiration date to one hour ago
    setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
    echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

**Output:**

Cookie 'user' is deleted.

### Advantages of Cookies :

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

### Disadvantages of Cookies:

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

## 4.6 SESSION

- A session is a way to store information (in variables) to be used across multiple pages. Sessions allow us to store data on the web server that associated with a session ID. Once we create a session, PHP sends a cookie that contains the session ID to the web browser.
- In the subsequent requests, the web browser sends the session ID cookie back to the web server so that PHP can retrieve the data based on the session ID and make the data available in our script.
- An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.
- A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.
- The location of the temporary file is determined by a setting in the php.ini file called session.save\_path.

### 4.6.1 Use of Session

- In general, session refers to a frame of communication between two medium. A PHP session is used to store data on a server rather than the computer of the user.
- Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc.
- A session is a global variable stored on the server. Each session is assigned a unique id which is used to retrieve stored values. Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server.
- When we work with an application, we open it, do some changes, and then we close it. This is much like a Session. The computer knows who we are. It knows when we start the application and when we end.
- But on the internet there is one problem, the web server does not know who we are or what we do, because the HTTP address does not maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc).
- By default, session variables last until the user closes the browser. So, session variables hold information about one single user, and are available to all pages in one application.

### Why should a session be maintained?

- When there is a series of continuous request and response from a same client to a server, the server cannot identify from which client it is getting requests.
- Because HTTP is a stateless protocol. When there is a need to maintain the conversational state, session tracking is needed.

### 4.6.2 Start Session

- A session is started with the `session_start()` function. The `session_start()` function first checks if a session is already started and if none is started then it starts one.
- Session variables are set with the PHP global variable `$_SESSION`. The `$_SESSION[ ]` variables can be accessed during lifetime of a session.

**Example:** For start session.

#### demo\_session1.php

```
<?php
    // Start the session
    session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
    // Set session variables
    $_SESSION["favcolor"] = "green";
    $_SESSION["favanimal"] = "cat";
    echo "Session variables are set.";
?>
</body>
</html>
```

**Output:**

Session variables are set.

- Make use of `isset()` function to check if session variable is already set or not.

### 4.6.3 Get Session Variables

- Next, we create another page called "demo\_session2.php". From this page, we will access the session information we set on the first page ("demo\_session1.php").
- Note that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (`session_start()`). Also notice that all session variable values are stored in the global `$_SESSION` variable.

**Example:** For get session variables.

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
    // echo session variables that were set on previous page
    echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";
    echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

**Output:**

Favorite color is green.  
Favorite animal is cat.

#### 4.6.4 Destroy Session

- To remove all global session variables and destroy the session, use `session_unset()` and `session_destroy()` functions.
- The `session_destroy()` function does not need any argument and a single call can destroy all the session variables.

**Example:** For destroy session.

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
    // remove all session variables
    session_unset();
    // destroy the session
    session_destroy();
?>
</body>
</html>
```

#### Difference between Session and Cookie:

Sr. No.	Cookie	Session
1.	Cookies are stored in browser as text file format.	Sessions are stored in server side.
2.	The cookie is a client-side resource.	The session is a server-side resource.
3.	It is stored limit amount of data.	It is stored unlimited amount of data.
4.	It is only allowing 4kb [4096bytes].	It is holding the multiple variable in sessions.
5.	It is not holding the multiple variable in cookies.	It is holding the multiple variable in sessions.
6.	We can accessing the cookies values in easily. So it is less secure.	We cannot accessing the session values in easily. So it is more secure.
7.	Setting the cookie time to expire the cookie.	using <code>session_destroy()</code> , we will destroyed the sessions.
8.	The <code>setcookie()</code> function must appear before the <code>&lt;html&gt;</code> tag.	The <code>session_start()</code> function must be the very first thing in your document. Before any HTML tags.
9.	Usually contains an id string.	Usually contains more complex information.
10.	Specific identifier links to server.	Specific identifier links to user.

#### 4.7 SENDING E-MAIL

- To send e-mail using PHP we must configure the `php.ini` file with the details of how the system sends e-mail.
- Open the file `php.ini` and go to the section entitled [mail function]. Set the SMTP setting:  
`smtp = smtp.my.server.net`
- Set the `sendmail_from` setting to reflect our e-mail address:  
`sendmail_from = abc@example.com`
- Linux users simply need to let PHP know the location of their `sendmail` application. The path and any desired switches should be specified to the `sendmail_path` directive.

- The configuration for Linux should look something like this:

```
smtp =
sendmail_from =
; forunix only
sendmail_path = /usr/sbin/sendmail -t -i
```

**Using the mail() Function:**

- PHP use mail() function to send an e-mail. This function requires three mandatory arguments that specify the recipient's e-mail address, the subject of the message and the actual message. Additionally there are other two optional parameters.

**Syntax:** bool mail(string \$to, string \$subject, string \$message [, string \$additional\_headers [, string \$additional\_parameters ]])

- Here, is the description for each parameters.

Sr. No	Parameters	Description
1.	to	Required. Specifies the receiver / receivers of the email.
2.	subject	Required. Specifies the subject of the email. This parameter cannot contain any newline characters.
3.	message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters.
4.	headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n).
5.	parameters	Optional. Specifies an additional parameter that can be used to pass additional flags as command line options to the program configured to be used when sending mail, as defined by the sendmail_path configuration setting.

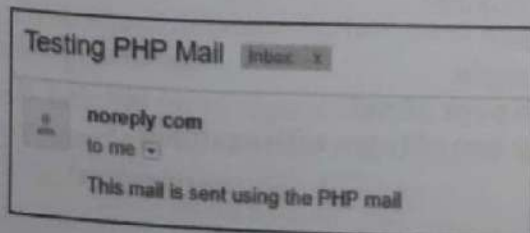
**Return Values:**

- Returns True if the mail was successfully accepted for delivery, False otherwise.
- Following example will send an email message to 'name @ company .com'.

**Example:**

```
<?php
Function sanitize_my_email($field) {
$field = filter_var($field, FILTER_SANITIZE_EMAIL);
if (filter_var($field, FILTER_VALIDATE_EMAIL)) {
return true;
} else {
return false;
}
}
$to_email = 'name @ company .com';
$subject = 'Testing PHP Mail';
$message = 'This mail is sent using the PHP mail ';
$headers = 'From: noreply @ company. com';
//check if the email address is invalid $secure_check
$secure_check = sanitize_my_email($to_email);
if ($secure_check == false) {
echo "Invalid input";
} else { //send email
mail($to_email, $subject, $message, $headers);
echo "This email is sent using PHP Mail";
}
?>
```

**Output:**



## How to configure XAMPP to send mail from localhost using PHP?

MercuryMail	31-08-2018 17:31	File folder	
mysql	31-08-2018 17:29	File folder	
perl	31-08-2018 17:30	File folder	
php	31-08-2018 17:31	File folder	contains php.ini
phpMyAdmin	01-09-2018 10:31	File folder	
sendmail	31-08-2018 17:29	File folder	contains sendmail.ini
src	31-08-2018 17:29	File folder	
tmp	05-12-2018 17:44	File folder	
tomcat	31-08-2018 17:29	File folder	
webalizer	31-08-2018 17:31	File folder	

**Step 1:** Go to C:\xampp\sendmail; open sendmail.ini file in notepad or any text editor and make the changes as follows:

```

smtp_server=mail.yourdomain.com → smtp_server=smtp.gmail.com
; smtp port (normally 25)
smtp_port → smtp_port=587
; SMTPS (SSL) support
; auto = use SSL for port 465, otherwise try to use TLS
; ssl = always use SSL
; tls = always use TLS
; none = never try to use SSL
smtp_ssl=auto → smtp_ssl=tls

```

**Step 2:** Go to C:\xampp\php; open php.ini file in notepad or any text editor goto [mail function] part and make the changes as follows.

```

[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP=localhost → ;SMTP=localhost
; http://php.net/smtp-port
smtp_port=25 → ;smtp_port=25

; For Win32 only.
; http://php.net/sendmail-from
sendmail_from = me@example.com → ;sendmail_from=me@example.com

; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
sendmail_path = → sendmail_path=C:\xampp\sendmail\sendmail.exe (Set the path where
the sendmail.exe)

```

### Practice Questions

1. What is form? How to create it?
2. What is form control?
3. Write short note on: Server role.
4. Explain processing forms with example.
5. When to use GET and POST method.
6. How to maintain state in PHP?
7. What is cookie? How to create it? Explain with example.
8. What is session? Explain with example.
9. How we can get the cookie values and destroy the cookies?
10. How to check whether a variable is set with a session?
11. What is form validation? Explain with suitable example.
12. Is it possible to create many forms using single web page. How?
13. Enlist different types of Form Controls. Explain any two of them with example.
14. How to send e-mail? Describe with example.

# 5...

## Database Operations

### Chapter Outcomes...

- Create database for the given problem using PHP script.
- Insert data in the given database using PHP script.
- Apply the specified update operation in database record using PHP script.
- Delete the given record from the database using PHP script.

### Learning Objectives...

- To understand Concept of Database and DBMS
- To study Basic Concepts in MySQL
- To learn Connectivity to MySQL Database
- To understand Database Operations

## 5.0 INTRODUCTION

- A database is a collection of related data. A database is a collection of information that is organized so that it can easily be accessed, managed and updated. With PHP, we can connect to and manipulate databases.
- A DataBase Management System (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.
- Nowadays, we use Relational Database Management Systems (RDBMS) to store and manage huge volume/amount of data. PHP will work with virtually all database software, including Oracle, PostgreSQL and Sybase but most commonly used is freely available MySQL database.

## 5.1 INTRODUCTION TO MySQL

- A PHP is very flexible with MySQL database compared with other databases. The data transfers between the PHP script and MySQL database very smoothly and flexibly.
- Nowadays MySQL is the most popular open source Relational Database Management System (RDBMS). The data in the MySQL database is stored in the form of tables.
- MySQL works very well in combination of various programming languages like PERL, C, C++, Java and PHP. Out of these languages, PHP is the most popular one because of its web application development capabilities.
- PHP provides various functions to access the MySQL database and to manipulate the data records inside the MySQL database.
- It is a web based database system used and runs on server. MySQL is published under an open-source license agreement.
- MySQL handles more expensive and powerful database packages. It uses standard SQL i.e statements or commands to be written will be same as SQL. It deals with C, C++, PHP, Java and so on and other working frameworks.

- MySQL works quick and handle even with tremendous information sets. It is neighborly to PHP, the most section on dialect for web improvement.

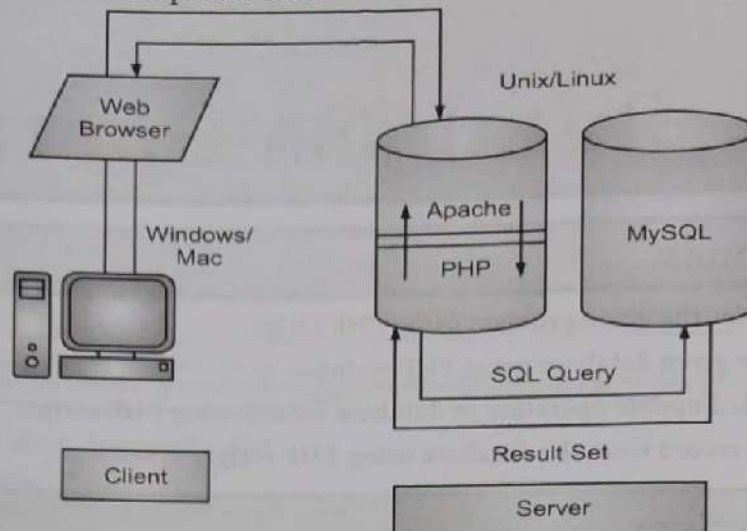


Fig. 5.1: Working of MySQL Database with PHP

### MySQL - PHP Syntax:

- PHP provides various functions to access the MySQL database and to manipulate the data records inside the MySQL database. We would require to call the PHP functions in the same way we call any other PHP function.
- The PHP functions for use with MySQL have the following general **format/syntax**:  
**Syntax:** `mysql_function(value, value, ...);`
- The second part of the function name is specific to the function, usually a word that describes what the function does. The following are two functions, which we will use in this chapter  
`mysqli_connect($connect);`  
`mysqli_query($connect, "SQL statement");`
- The following example shows a generic syntax of PHP to call any MySQL function.

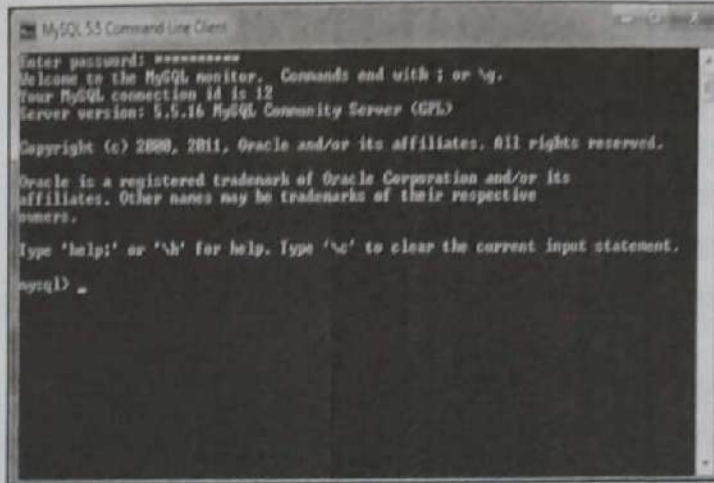
```
<html>
  <head>
    <title>PHP with MySQL</title>
  </head>
  <body>
    <?php
      $retval = mysql_function(value, [value,...]);
      if( !$retval )
      {
        die ( "Error: a related error message" );
      }
      // Otherwise MySQL or PHP Statements
    ?>
  </body>
</html>
```

- Databases are useful for storing information categorically. A company may have a database with the tables namely, Employees, Products, Customers and Orders.

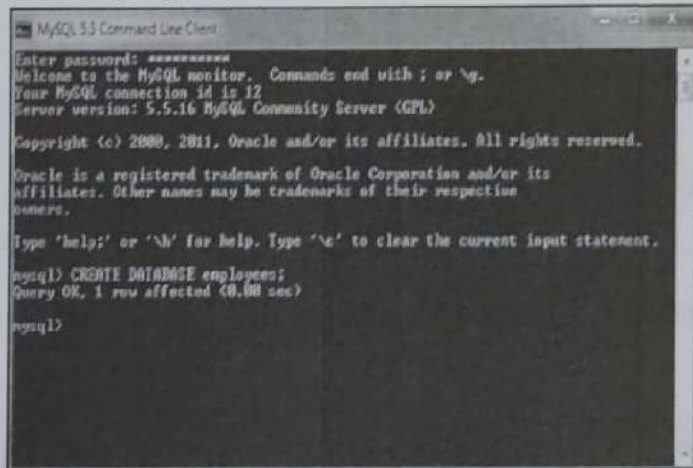


### 5.1.1 Create a Database

- A database is a collection of data. MySQL allows us to store and retrieve the data from the database in an efficient way.
- In MySQL, we can create a database using the CREATE DATABASE statement. But, if database already exists, it throws an error. To avoid the error, we can use the IF NOT EXISTS option with the CREATE DATABASE statement.
- We can create a MySQL database by using MySQL Command Line Client. Open the MySQL console and write down password, if we set one while installation. We will get the following:



- Now we are ready to create database.  
**Syntax:** CREATE DATABASE database\_name;  
**Example:** CREATE DATABASE employees;



- We can check the created database by the following query:  
**SHOW DATABASES;**  
**Output:**



**Note:** PHP also uses `mysql_query()` function to create a MySQL database. This function takes two parameters and returns True on success or False on failure.

**Syntax:** `bool mysql_query(sql, connection);`

PHP provides function `mysql_select_db()` to select a database. It returns True on success or False on failure.

**Syntax:** `bool mysql_select_db(db_name, connection);`

## 5.2 CONNECTING TO A MySQL DATABASE

- PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
- PHP 5 and later can work with a MySQL database using:
  1. MySQLi extension (the "i" stands for improved), and
  2. PDO (PHP Data Objects).

### 5.2.1 MySQL Database Server from PHP

- There are two functions of PHP to connect MySQL database namely, `mysqli_connect()` and `PDO::__construct()`.

#### 5.2.1.1 `mysqli_connect()` Function

- The `mysqli_connect()` function is used to connect with MySQL database. It returns resource if connection is established or null.

**Syntax:** `resource mysqli_connect(server, username, password)`

**Example:** For `welcome.html` file and it has a text field and submit button.

```
<?php
    $host = 'localhost:3306';
    $user = '';
    $pass = '';
    $conn = mysqli_connect($host, $user, $pass);
    if(! $conn )
    {
        die('Could not connect: ' . mysqli_error());
    }
    echo 'Connected successfully...';
    mysqli_close($conn);
?>
```

**Output:**

Connected successfully

#### 5.2.1.2 `PDO::__construct()` Function

- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
- So, if we have to switch our project to use another database, PDO makes the process easy. We only have to change the connection string and a few queries.

**Syntax:**

```
public PDO::__construct ( string $dsn [, string $username [, string $passwd [, array $options ]]] )
```

**Example:**

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

```

$password = "password";
try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully...";
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}
?>

```

**Output:**

Connected successfully...

**Note:** PHP provides `mysql_connect()` function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success or False on failure.

**Syntax:** `connection mysql_connect(server, user, passwd, new_link, client_flag);`

We can disconnect from the MySQL database anytime using another PHP function `mysql_close()`.

**Syntax:** `bool mysql_close(resource $link_identifier);`

If a resource is not specified, then the last opened database is closed. This function returns true if it closes the connection successfully otherwise it returns false.

The `CREATE TABLE` statement is used to create a table in MySQL. To create tables in the new database we need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using `mysql_query()` function.

## 5.3 DATABASE OPERATIONS

- With PHP, we can connect and manipulate databases. MySQL is the most popular database system used with PHP which includes operations such as insert data, delete data, retrieve data, update data and so on.
- PHP `mysql_query()` function is used to insert, select, delete and update record in a table.

### 5.3.1 Insert Data

- PHP `mysql_query()` function is used to insert record in a table. Data can be entered into MySQL tables by executing SQL `INSERT` statement through PHP function `mysql_query`.

**Example:** For insert data.

```

<?php
    $host = 'localhost:3306';
    $user = '';
    $pass = '';
    $dbname = 'test';
    $conn = mysqli_connect($host, $user, $pass,$dbname);
    if(!$conn)
    {
        die('Could not connect: '.mysqli_connect_error());
    }
    echo 'Connected successfully...<br/>';
    $sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
    if(mysqli_query($conn, $sql)){

```

```

    echo "Record inserted successfully...";
  }else{
    echo "Could not insert record: ". mysqli_error($conn);
  }
  mysqli_close($conn);
?>

```

**Output:**

```

Connected successfully...
Record inserted successfully...

```

**5.3.2 Retrieving the Query Result**

- PHP `mysql_query()` function is used to execute select query. Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function `mysql_query()`.
- There are two other MySQLi functions used in select query.
  1. **`mysqli_num_rows(mysqli_result $result)`** which returns number of rows.
  2. **`mysqli_fetch_assoc(mysqli_result $result)`** which returns row as an associative array. Each key of the array represents the column name of the table. It return NULL if there are no more rows.

**Example:** For retrieving data.

```

<?php
    $host = 'localhost:3306';
    $user = '';
    $pass = '';
    $dbname = 'test';
    $conn = mysqli_connect($host, $user, $pass,$dbname);
    if(!$conn){
        die('Could not connect: '.mysqli_connect_error());
    }
    echo 'Connected successfully...<br/>';
    $sql = 'SELECT * FROM emp4';
    $retval=mysqli_query($conn, $sql);
    if(mysqli_num_rows($retval) > 0){
        while($row = mysqli_fetch_assoc($retval)){
            echo "EMP ID :{$row['id']} <br> ".
                "EMP NAME : {$row['name']} <br> ".
                "EMP SALARY : {$row['salary']} <br> ".
                "-----<br>";
        } //end of while
    }else{
        echo "0 results";
    }
    mysqli_close($conn);
?>

```

**Output:**

```

Connected successfully...
EMP ID :1
EMP NAME :ratan
EMP SALARY : 9000
-----

```

```
EMP ID :2
EMP NAME :karan
EMP SALARY : 40000
-----
```

```
EMP ID :3
EMP NAME : jai
EMP SALARY : 90000
-----
```

## 5.4 UPDATE AND DELETE OPERATIONS ON TABLE DATA

- Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function `mysql_query()`.
- Data can be deleted from MySQL tables by executing SQL DELETE statement through PHP function `mysql_query()`.

### 5.4.1 Update Data

- PHP `mysql_query()` function is used to update record in a table.

**Example:** For update data.

```
<?php
    $host = 'localhost:3306';
    $user = '';
    $pass = '';
    $dbname = 'test';
    $conn = mysqli_connect($host, $user, $pass,$dbname);
    if(!$conn){
        die('Could not connect: '.mysqli_connect_error());
    }
    echo 'Connected successfully...<br/>';
    $id=2;
    $name="Rahul";
    $salary=80000;
    $sql = "update emp4 set name=\"\$name\", salary=\$salary where id=\$id";
    if(mysqli_query($conn, $sql)){
        echo "Record updated successfully...";
    }else{
        echo "Could not update record: ". mysqli_error($conn);
    }
    mysqli_close($conn);
?>
```

**Output:**

```
Connected successfully...
Record updated successfully...
```

### 5.4.2 Delete Data

- PHP `mysql_query()` function is used to delete record in a table.

**Example:** For delete data.

```
<?php
    $host = 'localhost:3306';
    $user = '';
    $pass = '';
```

```
$dbname = 'test';
$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully...<br/>';
$id=2;
$sql = "delete from emp4 where id=$id";
if(mysqli_query($conn, $sql)){
echo "Record deleted successfully...";
}else{
echo "Could not deleted record: ". mysqli_error($conn);
}
mysqli_close($conn);
?>
```

**Output:**

Connected successfully...

Record deleted successfully...

**Practice Questions**

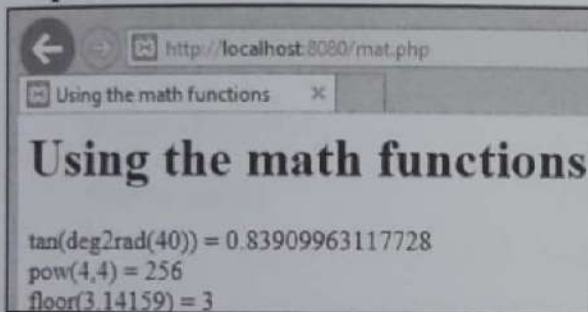
1. What is database?
2. What is DBMS?
3. What is MySQL? How it is used in PHP?
4. How to create and delete a database in MySQL?
5. Explain `mysqli_connect()` function with example.
6. Explain `PDO::__construct()` function with example.
7. Write a program to create an employee table to perform insert, delete and update operations.
8. How to connect database to MySQL? Explain with example.
9. How to insert a record in table in MySQL with PHP?
10. How to update a record in table in MySQL with PHP?
11. How to retrieve a record in table in MySQL with PHP?

# Programs

## 1. Program for Math function.

```
<html>
  <head>
    <title> Using the math functions </title>
  </head>
  <body>
  <h1> Using the math functions </h1>
  <?php
    echo "tan(deg2rad(40)) = ", tan(deg2rad(40)), "<br>";
    echo "pow(4,4) = ", pow(4,4), "<br>";
    echo "floor(3.14159) = ", floor(3.14159), "<br>";
  ?>
</body>
</html>
```

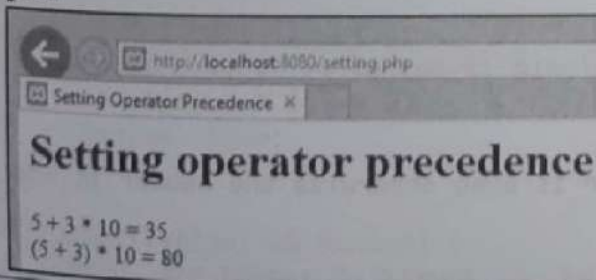
### Output:



## 2. Program for setting operator precedence.

```
<html>
  <head>
    <title>Setting Operator Precedence</title>
  </head>
  <body>
  <h1> Setting operator precedence </h1>
  <?php
    echo "5 + 3 * 10 = ", 5 + 3 * 10, "<br>";
    echo "(5 + 3) * 10 = ", (5 + 3) * 10, "<br>";
  ?>
</body>
</html>
```

### Output:



## 3. Program to find sum of digits of a number.

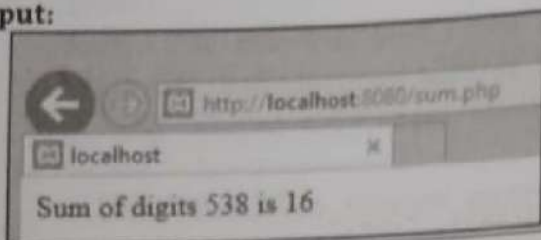
```
<?php
  $num = 538;
  $sum=0; $rem=0;
```

```

for ($i = 0; $i <= strlen($num); $i++)
{
    $rem = $num % 10;
    $sum = $sum + $rem;
    $num = $num / 10;
}
echo "Sum of digits 538 is $sum";
?>

```

Output:



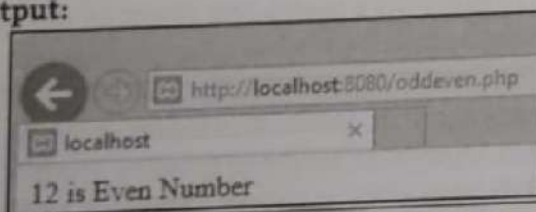
#### 4. Program to check given number is odd or even.

```

<?php
    $number = 12;
    if ($number % 2 == 0)
    {
        echo "$number is Even Number";
    }
    else
    {
        echo "$number is Odd Number";
    }
?>

```

Output:



#### 5. Program to check given number is odd or even input is taken from user.

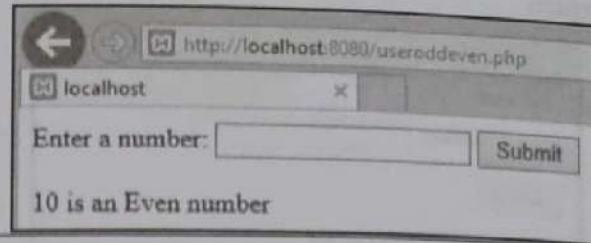
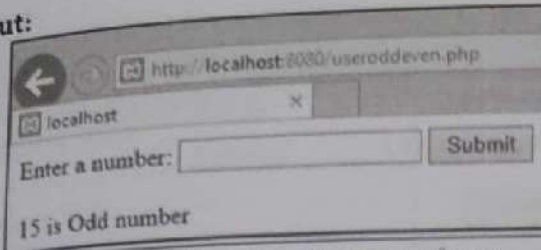
```

<html>
<body>
<form method="post">
    Enter a number:
<input type="number" name="number">
<input type="submit" value="Submit">
</form>
</body>
</html>
<?php
if($_POST){
    $number = $_POST['number'];
    //divide entered number by 2
    //if the remainder is 0 then the number is even otherwise the number is odd
    if(($number % 2) == 0){
        echo "$number is an Even number";
    }else{
        echo "$number is Odd number";
    }
}
?>

```



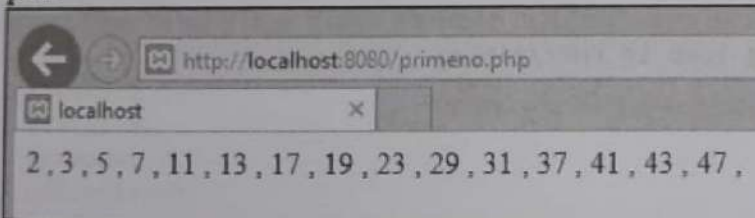
Output:



6. Program to list the first 15 prime numbers.

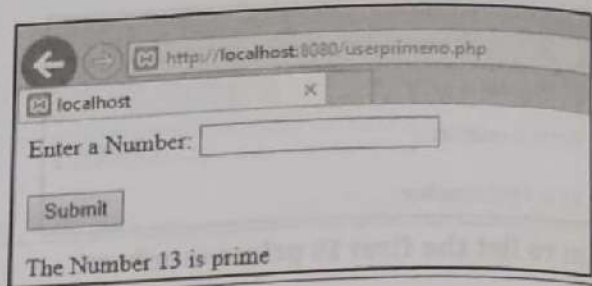
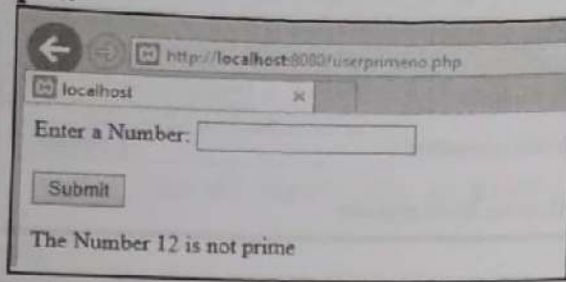
```
<?php
$count = 0;
$num = 2;
while ($count < 15 )
{
    $div_count=0;
    for ( $i=1; $i<=$num; $i++)
    {
        if (($num%$i)==0)
        {
            $div_count++;
        }
    }
    if ($div_count<3)
    {
        echo $num." , ";
        $count=$count+1;
    }
    $num=$num+1;
}
?>
```

Output:



7. Program which will check whether a number is prime or not.

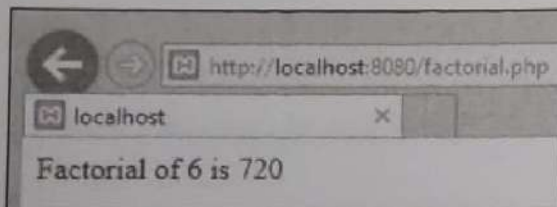
```
<form method="post">
Enter a Number: <input type="text" name="input"><br><br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
if($_POST)
{
    $input=$_POST['input'];
    for ($i = 2; $i <= $input-1; $i++) {
        if ($input % $i == 0) {
            $value= True;
        }
    }
}
if (isset($value) && $value) {
    echo 'The Number ' . $input . ' is not prime';
} else {
    echo 'The Number ' . $input . ' is prime';
}
?>
```

**Output:****8. Program to print factorial of a number.**

```

<?php
    $num = 6;
    $factorial = 1;
    for ($x=$num; $x>=1; $x--)
    {
        $factorial = $factorial * $x;
    }
    echo "Factorial of $num is $factorial";
?>

```

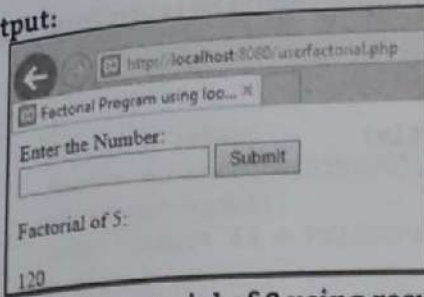
**Output:****9. Program shows a form through which we can calculate factorial of any number.**

```

<html>
<head>
<title>Factorial Program using loop in PHP</title>
</head>
<body>
<form method="post">
    Enter the Number:<br>
<input type="number" name="number" id="number">
<input type="submit" name="submit" value="Submit" />
</form>
<?php
if($_POST){
    $fact = 1;
    //getting value from input text box 'number'
    $number = $_POST['number'];
    echo "Factorial of $number:<br><br>";
    //start loop
    for ($i = 1; $i<= $number; $i++)
    {
        $fact = $fact * $i;
    }
    echo $fact . "<br>";
}
?>
</body>
</html>

```

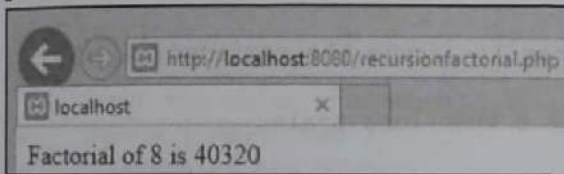
Output:



### 10. Program to Factorial of 8 using recursion method.

```
<?php
function fact ($n)
{
    if($n <= 1)
    {
        return 1;
    }
    else
    {
        return $n * fact($n - 1);
    }
}
echo "Factorial of 8 is " .fact(8);
?>
```

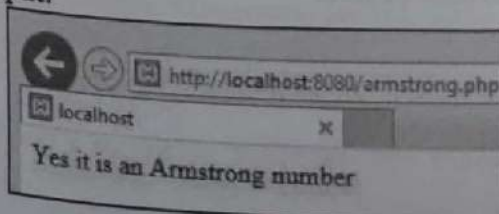
Output:



### 11. Program checks whether given number is Armstrong or not.

```
<?php
$num=371;
$total=0;
$x=$num;
while($x!=0)
{
    $rem=$x%10;
    $total=$total+$rem*$rem*$rem;
    $x=$x/10;
}
if($num==$total)
{
    echo "Yes it is an Armstrong number";
}
else
{
    echo "No it is not an armstrong number";
}
?>
```

Output:



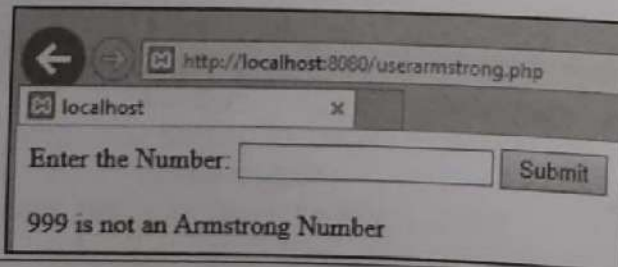
## 12. Program checks whether given number is Armstrong or not input should be taken from user.

```

<html>
  <head>
    <title> To find number is Armstrong or not </title>
  </head>
  <body>
    <form method = "post">
      Enter the Number: <br>
      <input type = "number" name = "number">
      <input type = "submit" value = "submit">
    </form>
  </body>
</html>
<?php
if($_POST)
{
  //get the number entered
  $number = $_POST['number'];
  //store entered number in a variable
  $a = $number;
  $sum = 0;
  //run loop till the quotient is 0
  while( $a != 0 )
  {
    $rem = $a % 10; //find remainder
    $sum = $sum + ( $rem * $rem * $rem ); //cube the remainder and add it to the
    sum variable till the loop ends
    $a = $a / 10; //find quotient. if 0 then loop again
  }
  //if the entered number and $sum value matches then it is an armstrong number
  if( $number == $sum )
  {
    echo "Yes $number an Armstrong Number";
  }else
  {
    echo "$number is not an Armstrong Number";
  }
}
</body>
</html>

```

### Output:



## 13. Program to check whether the given number is Palindrome or not.

```

<?php
function palindrome($n){
  $number = $n;
  $sum = 0;
  while(floor($number)) {
    $rem = $number % 10;
    $sum = $sum * 10 + $rem;

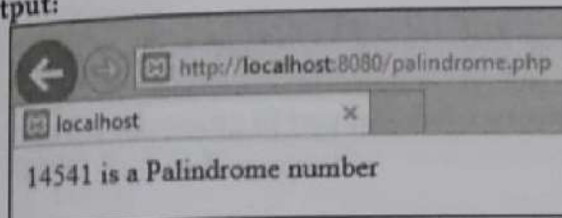
```

```

    $number = $number/10;
  }
  return $sum;
}
$input = 14541;
$num = palindrome($input);
if($input==$num){
echo "$input is a Palindrome number";
} else {
echo "$input is not a Palindrome";
}
}
?>

```

Output:



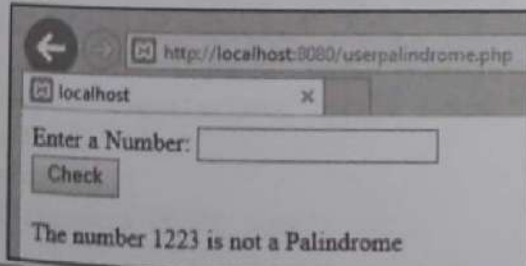
#### 14. Program to check whether the enter number is Palindrome or not (input from the user).

```

<form method="post">
Enter a Number: <input type="text" name="num"/><br>
<button type="submit">Check</button>
</form>
<?php
if($_POST)
{
    //get the value from form
    $num = $_POST['num'];
    //reversing the number
    $reverse = strrev($num);
    //checking if the number and reverse is equal
    if($num == $reverse){
    echo "The number $num is Palindrome";
    }else{
    echo "The number $num is not a Palindrome";
    }
}
?>

```

Output:



#### 15. Program to print the first 12 numbers of a Fibonacci series.

```

<?php
/* Print fiboancci series upto 12 elements. */
$num = 12;
echo "<h3>Fibonacci series using recursive function:</h3>";
echo "\n";
/* Recursive function for fibonacci series. */
function series($num){
if($num == 0){

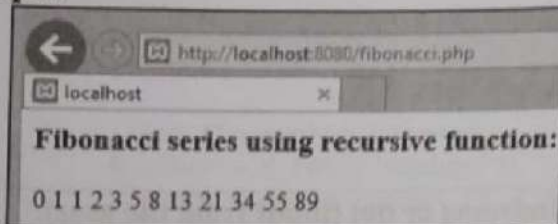
```

```

return 0;
}else if( $num == 1){
return 1;
} else {
return (series($num-1) + series($num-2));
}
}
/* Call Function. */
for ($i = 0; $i < $num; $i++){
echo series($i);
echo "\n";
}
?>

```

Output:



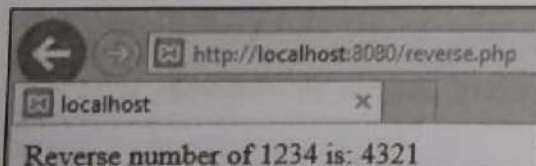
### 16. Program to find reverse of a number.

```

<?php
$num = 1234;
$revnum = 0;
while ($num > 1)
{
$rem = $num % 10;
$revnum = ($revnum * 10) + $rem;
$num = ($num / 10);
}
echo "Reverse number of 1234 is: $revnum";
?>

```

Output:



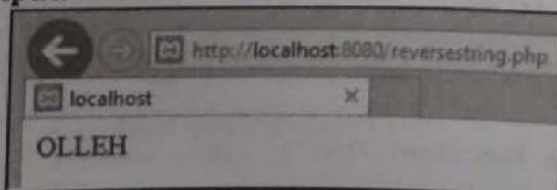
### 17. Program to reverse the string.

```

<?php
$string = "HELLO";
$length = strlen($string);
for ($i=($length-1) ; $i >= 0 ; $i--)
{
echo $string[$i];
}
?>

```

Output:



### 18. Program for swapping of two numbers.

```

<?php
$a = 16;

```

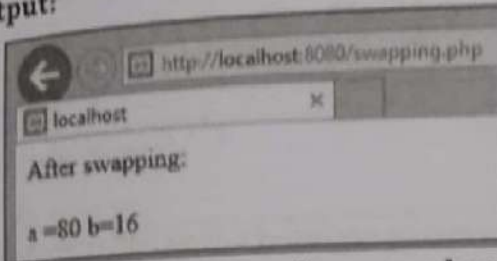
```

$b = 80;
// Swapping Logic
$third = $a;
$a = $b;
$b = $third;
echo "After swapping:<br><br>";
echo "a = ".$a." b=".$b;

```

?>

Output:



### 19. Program for swapping of two numbers without using third variable.

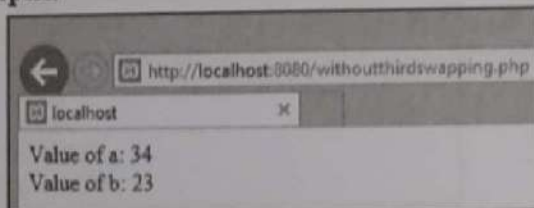
```

<?php
$a=23;
$b=34;
//using arithmetic operation
$a=$a+$b;
$b=$a-$b;
$a=$a-$b;
echo "Value of a: $a</br>";
echo "Value of b: $b</br>";

```

?>

Output:

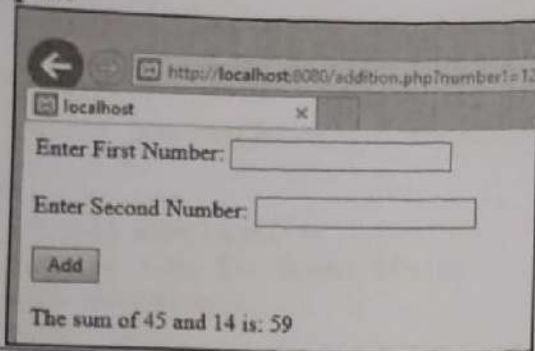


### 20. Program for addition of two numbers.

```

<html>
<body>
<form method="post">
    Enter First Number:
    <input type="number" name="number1" /><br><br>
    Enter Second Number:
    <input type="number" name="number2" /><br><br>
    <input type="submit" name="submit" value="Add">
</form>
<?php
if(isset($_POST['submit']))
{
    $number1 = $_POST['number1'];
    $number2 = $_POST['number2'];
    $sum = $number1+$number2;
    echo "The sum of $number1 and $number2 is: ".$sum;
}
?>
</body>
</html>

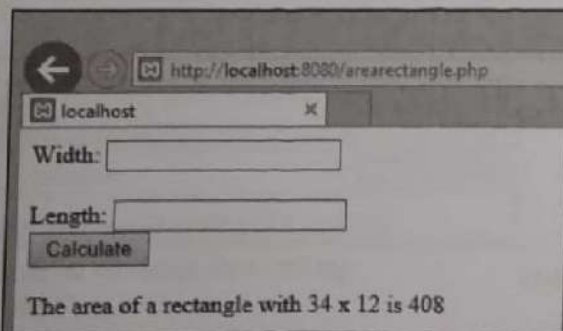
```

**Output:****21. Program to calculate area of rectangle.**

```

<html>
<body>
<form method = "post">
Width: <input type="number" name="width">
<br><br>
Length: <input type="number" name="length"><br>
<input type = "submit" name = "submit" value="Calculate">
</form>
</body>
</html>
<?php
if(isset($_POST['submit']))
{
$width = $_POST['width'];
$length = $_POST['length'];
$area = $width*$length;
    echo "The area of a rectangle with $width x $length is $area";
}
?>

```

**Output:****22. Print the following triangle pattern:**

AAAAA

BBBBB

CCC

DD

E

```

<?php
$alpha = range('A', 'Z');
for($i=0; $i<5; $i++){
for($j=5; $j>$i; $j--){
echo $alpha[$i];
}
echo "<br>";
}
}
?>

```

?&gt;



Web Based Application Development with PHP

23. Print the following triangle pattern:

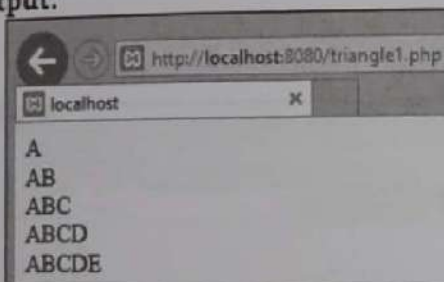
A  
BB  
CCC  
DDDD  
EEEE.

```
<?php
$alpha = range('A', 'Z');
for($i=0; $i<5; $i++){
for($j=0; $j<=$i; $j++){
echo $alpha[$i];
}
echo "<br>";
}
?>
```

24. Program to print the triangle pattern:

```
<?php
$alpha = range('A', 'Z');
for($i=0; $i<5; $i++){
for($j=0; $j<=$i; $j++){
echo $alpha[$j];
}
echo "<br>";
}
?>
```

Output:

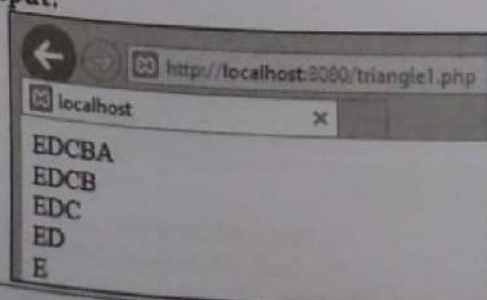


25. Program to print the following triangle pattern.

EDCBA  
EDCB  
EDC  
ED  
E

```
<?php
$alpha = range('A', 'Z');
for($i=0; $i<5; $i++){
for($j=4; $j>=$i; $j--){
echo $alpha[$j];
}
echo "<br>";
}
?>
```

Output:



26. Program to print the following triangle pattern.

```
5 5 5 5
4 4 4 4
3 3 3
2 2
1.
```

```
<?php
    for($i=5;$i>=1;$i--){
        for($j=$i;$j>=1;$j--){
            echo $i." ";
        }
        echo "<br>";
    }
?>
```

27. Program to print the following triangle pattern:

```
54321
4321
321
21
1
```

```
<?php
    for($i=0;$i<=5;$i++){
        for($j=5-$i;$j>=1;$j--){
            echo $j;
        }
        echo "<br>";
    }
?>
```

28. Program to print the following triangle pattern.

```
11111
1111
111
11
1
```

```
<?php
    for($i=0;$i<=5;$i++){
        for($j=5-$i;$j>=1;$j--){
            echo "1";
        }
        echo "<br>";
    }
?>
```

29. Program to print the following triangle pattern.

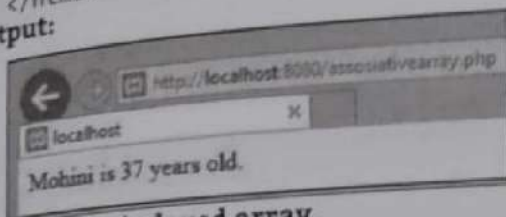
```
1
2 3
4 5 6
7 8 9 10
```

```
<?php
    $k=1;
    for($i=0;$i<4;$i++){
        for($j=0;$j<=$i;$j++){
            echo $k." ";
            $k++;
        }
        echo "<br>";
    }
?>
```

### 30. Program for associative array.

```
<html>
<body>
<?php
    $age=array("Ram"=>"35", "Mohini"=>"37", "Rahul"=>"43");
    echo "Mohini is " . $age['Mohini'] . " years old.";
?>
</body>
</html>
```

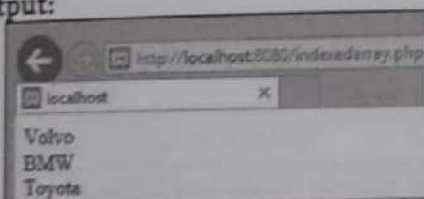
Output:



### 31. Program for indexed array.

```
<html>
<body>
<?php
    $cars=array("Volvo", "BMW", "Toyota");
    $arrlength=count($cars);
    for($x=0;$x<$arrlength;$x++)
    {
        echo $cars[$x];
        echo "<br>";
    }
?>
</body>
</html>
```

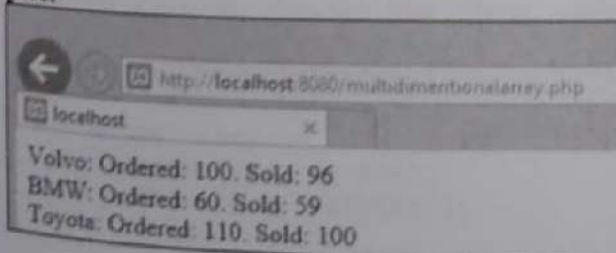
Output:



### 32. Program for multidimensional array.

```
<html>
<body>
<?php
    // A two-dimensional array
    $cars=array
    (
        array("Volvo",100,96),
        array("BMW",60,59),
        array("Toyota",110,100)
    );
    echo $cars[0][0].": Ordered: ".$cars[0][1].". Sold: ".$cars[0][2]."<br>";
    echo $cars[1][0].": Ordered: ".$cars[1][1].". Sold: ".$cars[1][2]."<br>";
    echo $cars[2][0].": Ordered: ".$cars[2][1].". Sold: ".$cars[2][2]."<br>";
?>
</body>
</html>
```

Output:

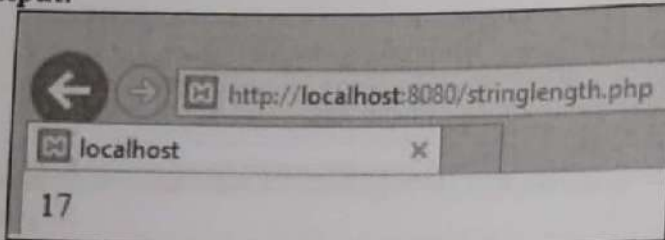


**33. Program to calculate the length of the string.**

```

<?php
// PHP program to find the
// length of a given string
$str = "A vanishPokharkar";
// prints the length of the string
// including the space
echo strlen($str);
?>

```

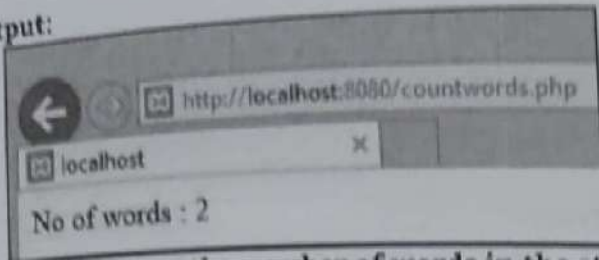
**Output:****34. Program to count the number of words in the string without using the string function.**

```

<?php
// PHP program to count no of
// words from given input string
$OUT = 0;
$IN = 1;
// returns number of words in str
functioncountWords($str)
{
global $OUT, $IN;
    $state = $OUT;
    $wc = 0; // word count
    $i = 0;
    // Scan all characters one by one
while ($i<strlen($str))
    {
        // If next character is
        // a separator, set the
        // state as OUT
if ($str[$i] == " " ||
    $str[$i] == "\n" ||
    $str[$i] == "\t")
        $state = $OUT;
        // If next character is not a
        // word separator and state is
        // OUT, then set the state as
        // IN and increment word count
else if ($state == $OUT)
        {
            $state = $IN;
            ++$wc;
        }
        // Move to next character
        ++$i;
    }
return $wc;
}
// Driver Code
$str = "AvanishPokharkar";
echo "No of words : " . countWords($str);
// This code is contributed
// by ChitraNayal
?>

```

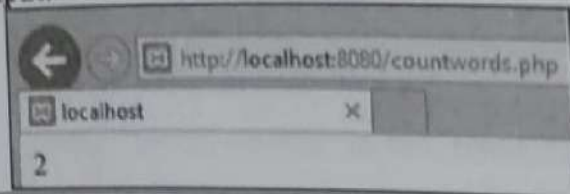
Output:



35. Program to count the number of words in the string with string function.

```
<?php
    $my_str = 'Good Morning.';
    echostr_word_count($my_str);
?>
```

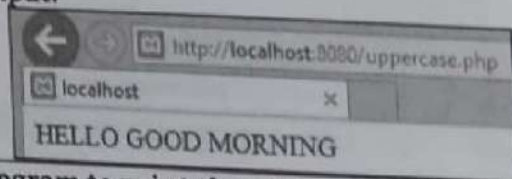
Output:



36. Program to print the given string in uppercase.

```
<?php
    # PHP code to convert to Upper Case
    functiontoUpper($string){
        return(strtoupper($string));
    }
    // Driver Code
    $string="Hello Good Morning";
    echo (toUpper($string));
?>
```

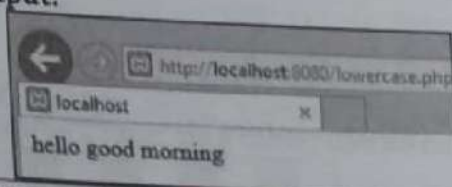
Output:



37. Program to print the given string in the lowercase.

```
<?php
    # PHP code to convert to Lower Case
    functiontoLower($string){
        return(strtolower($string));
    }
    // Driver Code
    $string="Hello Good Morning";
    echo (toLower($string));
?>
```

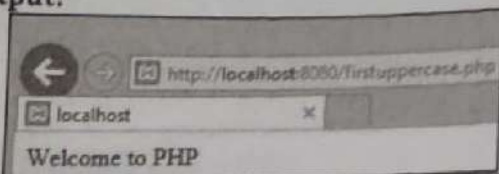
Output:



38. Program to convert the first letter in the uppercase.

```
<?php
    # PHP code to convert the first letter to Upper Case
    functionfirstUpper($string){
        return(ucfirst($string));
    }
    // Driver Code
    $string="welcome to PHP";
    echo (firstUpper($string));
?>
```

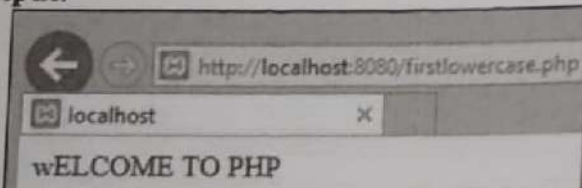
Output:



39. Program to convert the first letter in the lowercase.

```
<?php
# PHP code to convert the first letter to Lower Case
functionfirstLower($string){
return(lcfirst($string));
}
// Driver Code
$string="WELCOME TO PHP";
echo (firstLower($string));
?>
```

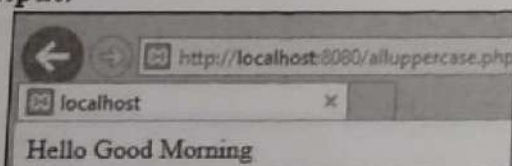
Output:



40. Program to convert the first letter of each word to Upper Case.

```
<?php
functionfirstUpper($string){
return(ucwords($string));
}
// Driver Code
$string="hello good morning";
echo (firstUpper($string));
```

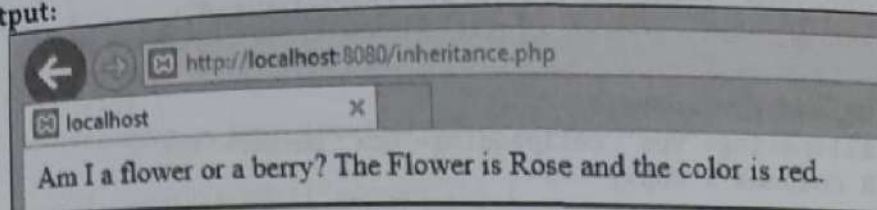
Output:



41. Program for inheritance.

```
<html>
<body>
<?php
class Flower {
public $name;
public $color;
public function __construct($name, $color) {
$this->name = $name;
$this->color = $color;
}
protected function intro() {
echo "The Flower is {$this->name} and the color is {$this->color}.";
}
}
class Rose extends Flower {
public function message() {
echo "Am I a flower or a berry? ";
// Call protected function from within derived class - OK
$this->intro();
}
}
$rose = new Rose("Rose", "red"); // OK. __construct() is public
$rose->message(); // OK. message() is public and it calls intro() (which is
protected) from within the derived class
```

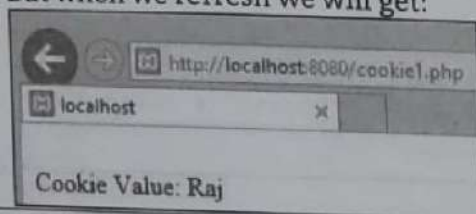
```
?>
</body>
</html>
```

**Output:****42. PHP cookie program.**

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

**Output:**

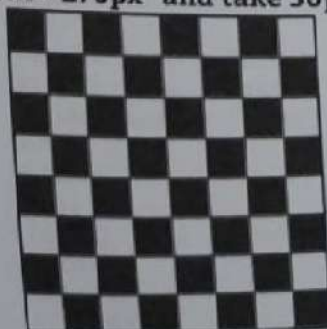
First we run it we will get output as sorry, cookie is not found.  
But when we refresh we will get:

**43. PHP script to lower-case and upper-case, all elements in an array.**

```
<?php
$colors = array( "Red", "Green", "Black", "White");
print_r($colors);
echo "<br>";
$lower_colors = array_map('strtolower', $colors);
print_r($lower_colors);
echo "<br>";
$upper_colors = array_map('strtoupper', $colors);
print_r($upper_colors);
?>
```

**Output:**

```
Array ([0] => Red [1] => Green [2] => Black [3] => White)
Array ([0] => red [1] => green [2] => black [3] => white)
Array ([0] => RED [1] => GREEN [2] => BLACK [3] => WHITE)
```

**44. PHP script using nested for loop that creates a chess board as shown below. Use table width="270px" and take 30px as cell height and width.**

```

<html>
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h3>Chess Board using Nested For Loop</h3>
<table width="270px" cellspacing="0px" cellpadding="0px" border="1px">
<!-- cell 270px wide (8 columns x 60px) -->
<?php
    for($row=1;$row<=8;$row++)
    {
    echo "<tr>";
    for($col=1;$col<=8;$col++)
    {
    $total=$row+$col;
    if($total%2==0)
    {
    echo "<td height=30px width=30px bgcolor= #FFFFFF></td>";
    }
    else
    {
    echo "<td height=30px width=30px bgcolor=#000000></td>";
    }
    }
    echo "</tr>";
    }
?>
</table>
</body>
</html>

```

#### 45. Develop a web page for the form validation in PHP.

```

<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
    // define variables and set to empty values
    $nameErr = $emailErr = $genderErr = $websiteErr = "";
    $name = $email = $gender = $comment = $website = "";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
    $nameErr = "Name is required";
    }else {
    $name = test_input($_POST["name"]);
    }
    if (empty($_POST["email"])) {
    $emailErr = "Email is required";
    }else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid email format";
    }
    }
    if (empty($_POST["website"])) {

```



```

$website = "";
}else {
$website = test_input($_POST["website"]);
}
if (empty($_POST["comment"])) {
$comment = "";
}else {
$comment = test_input($_POST["comment"]);
}
if (empty($_POST["gender"])) {
$genderErr = "Gender is required";
}else {
$gender = test_input($_POST["gender"]);
}
}
function test_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}
?>
<h2>Absolute classes registration</h2>
<p><span class = "error">* required field.</span></p>
<form method = "post" action = "<?php
echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
<table>
<tr>
<td>Name:</td>
<td><input type = "text" name = "name">
<span class = "error">* <?php echo $nameErr;?></span>
</td>
</tr>
<tr>
<td>E-mail: </td>
<td><input type = "text" name = "email">
<span class = "error">* <?php echo $emailErr;?></span>
</td>
</tr>
<tr>
<td>Time:</td>
<td><input type = "text" name = "website">
<span class = "error"><?php echo $websiteErr;?></span>
</td>
</tr>
<tr>
<td>Classes:</td>
<td><textarea name = "comment" rows = "5" cols = "40"></textarea></td>
</tr>
<tr>
<td>Gender:</td>
<td>
<input type = "radio" name = "gender" value = "female">Female
<input type = "radio" name = "gender" value = "male">Male
<span class = "error">* <?php echo $genderErr;?></span>
</td>
</tr>
<tr>
<td>
<input type = "submit" name = "submit" value = "Submit">
</td>
</tr>
</table>
</form>

```

```

<?php
echo "<h2>Your given values are as:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>

```

**Output:**

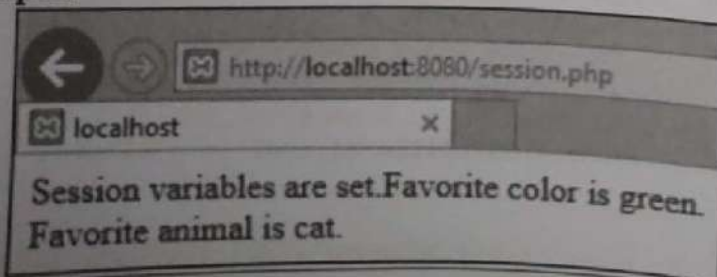
A screenshot of a web browser window showing a registration form. The browser's address bar displays 'http://localhost:8080/webpage.php'. The page title is 'Absolute classes registration'. The form contains several input fields: 'Name', 'E-mail', and 'Time', each followed by an asterisk indicating it is a required field. There is a larger text area for 'Classes'. Below these is a 'Gender' section with radio buttons for 'Female' and 'Male'. A 'Submit' button is located at the bottom left of the form. Below the form, the text 'Your given values are as:' is visible.

**46. Program for session management.**

```

<?php
    // Start the session
    session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
    // Set session variables
    $_SESSION["favcolor"] = "green";
    $_SESSION["favanimal"] = "cat";
    echo "Session variables are set.";
    echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";
    echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>

```

**Output:**

## 47. Program for mail function.

```

<html>
<head>
<title>Sending HTML email using PHP</title>
</head>
<body>
<?php
    $to = "xyz@somedomain.com";
    $subject = "This is subject";
    $message = "<b>This is HTML message.</b>";
    $message .= "<h1>This is headline.</h1>";
    $header = "From:abc@somedomain.com \r\n";
    $header .= "Cc:afgh@somedomain.com \r\n";
    $header .= "MIME-Version: 1.0\r\n";
    $header .= "Content-type: text/html\r\n";
    $retval = mail ($to,$subject,$message,$header);
    if( $retval == true ) {
    echo "Message sent successfully...";
    }else {
    echo "Message could not be sent...";
    }
?>
</body>
</html>

```

**Output:**

Message sent successfully...

## 48. Program to check that emails are valid.

```

<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
            $nameErr = "Only letters and white space allowed";
        }
    }
    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }
    if (empty($_POST["website"])) {
        $website = "";
    } else {

```

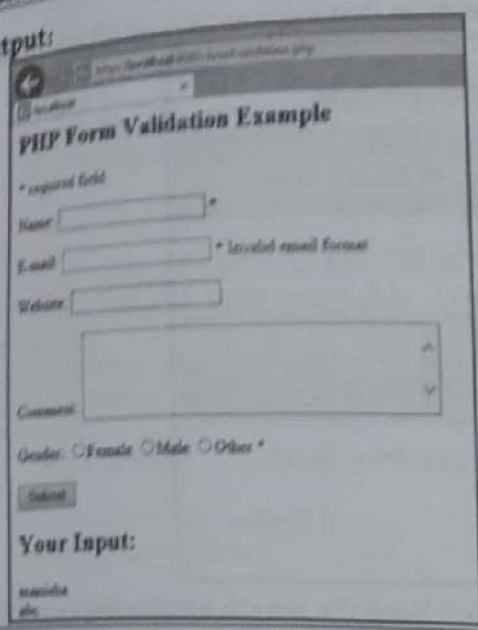


```

$website = test_input($_POST["website"]);
// check if URL address syntax is valid
if (!preg_match("/\b(?:https?|ftp):\/\/\w[\.\-a-z0-9+&@#\/%?~_!|:,;]*[\.\-a-z0-9+&@#\/%?~_!|/1", $website)) {
    $websiteErr = "Invalid URL";
}
}
if (empty($_POST["comment"])) {
    $comment = "";
} else {
    $comment = test_input($_POST["comment"]);
}
if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
} else {
    $gender = test_input($_POST["gender"]);
}
}
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    Name: <input type="text" name="name">
    <span class="error">* <?php echo $nameErr;?></span>
    <br><br>
    E-mail: <input type="text" name="email">
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    Website: <input type="text" name="website">
    <span class="error"><?php echo $websiteErr;?></span>
    <br><br>
    Comment: <textarea name="comment" rows="5" cols="40"></textarea>
    <br><br>
    Gender:
    <input type="radio" name="gender" value="female">Female
    <input type="radio" name="gender" value="male">Male
    <input type="radio" name="gender" value="other">Other
    <span class="error">* <?php echo $genderErr;?></span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>

```

Output:



49. Develop a simple application to enter data in database.

1. Creating PHP file.

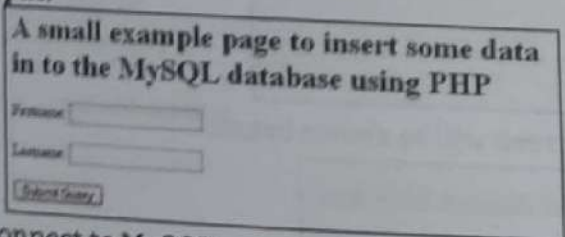
INDEX.PHP

```

<html>
<body>
<h1>A small example page to insert some data in to the MySQL database using PHP</h1>
<form action="insert.php" method="post">
Firstname: <input type="text" name="fname" /><br><br>
Lastname: <input type="text" name="lname" /><br><br>
<input type="submit" />
</form>
</body>
</html>

```

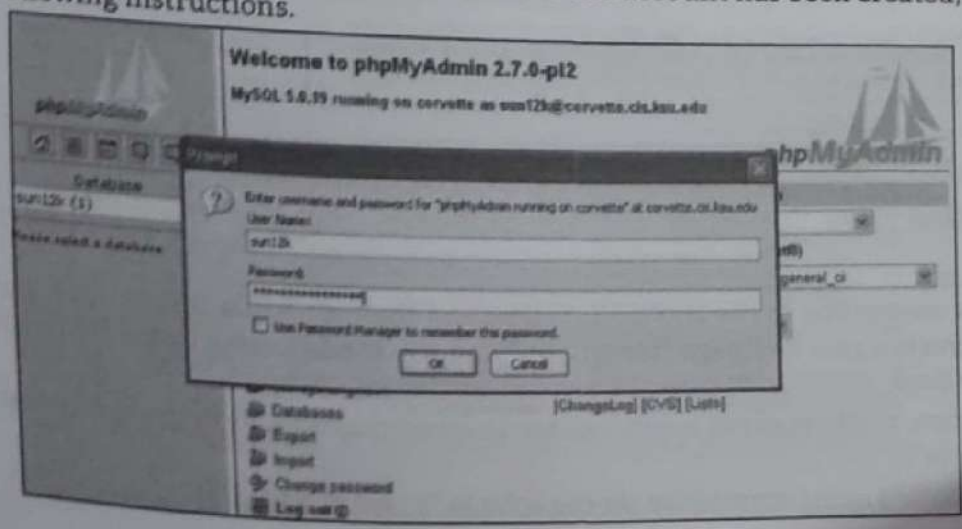
Output:



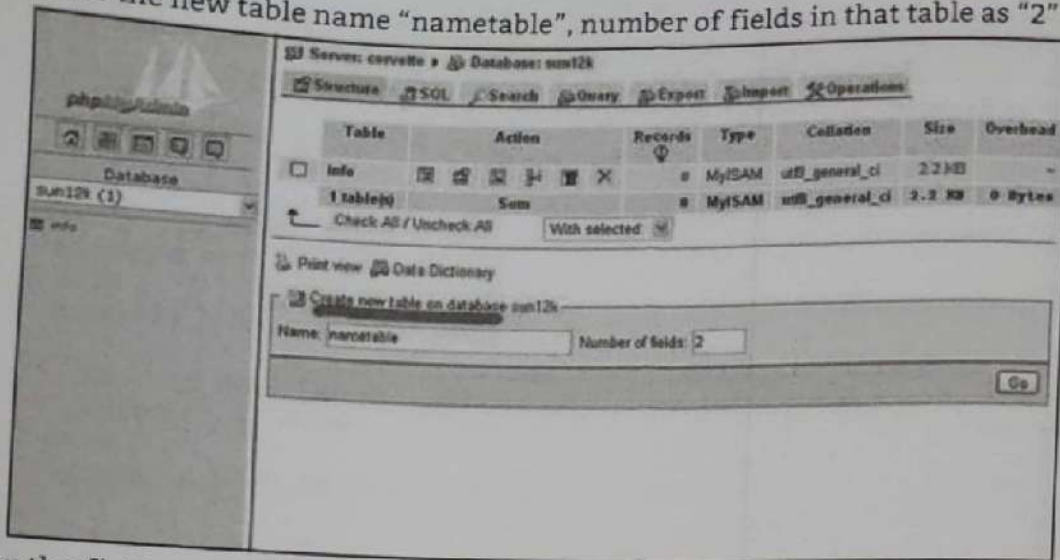
2. To connect to MySQL:

Before we can access your MySQL database, we must contact the system administrators to request an account.

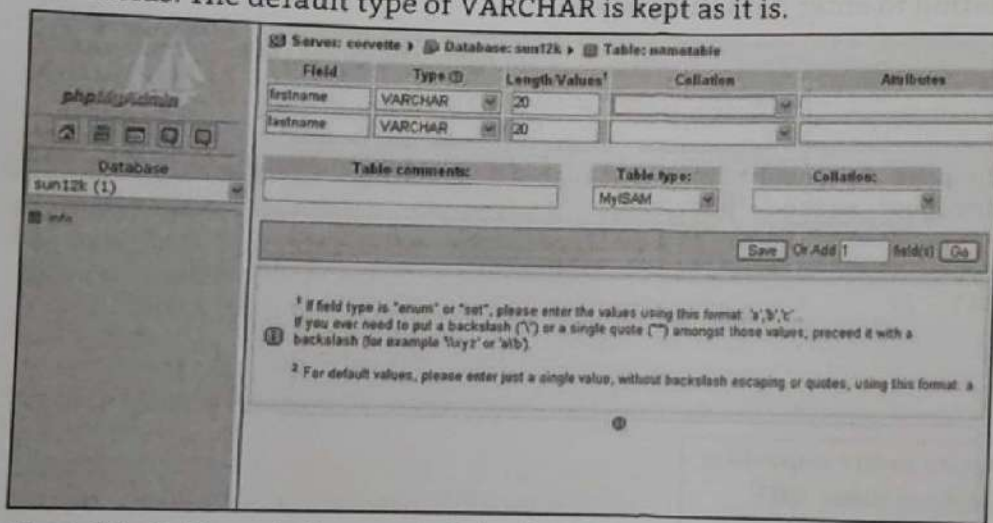
Once the administrators have notified we that the account has been created, we may connect using the following instructions.



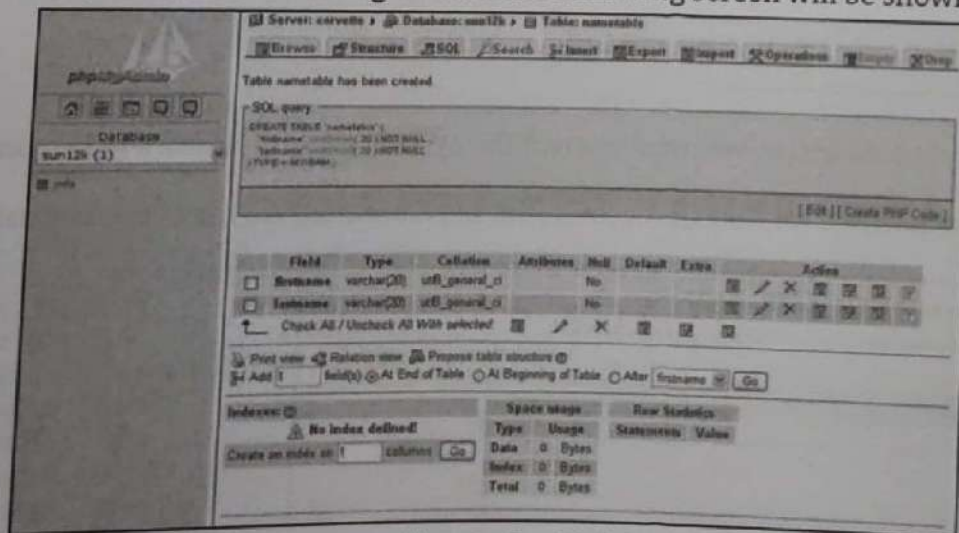
3. Now enter the new table name "nametable", number of fields in that table as "2" and hit Go button.



4. Enter the field names to be "firstname" and "lastname" and keep the length attributes to be "20" for both the fields. The default type of VARCHAR is kept as it is.



5. After the table field are being created the following screen will be shown below:



6. Now we need to make a connection to the MySQL database and then send this entered data from our textboxes. For that we create a new PHP page "insert.php" and use the following connection strings to the connection variable \$con  
 After making a connection, a SQL query is being written to enter this data in to the MySQL database being created ("nametable")  
 To tell the user that the data is being entered we set the echo to "1 record added".

### INSERT.PHP

```

<body>
<?php
$con = mysql_connect("mysql.cis.ksu.edu","cis_id","password");
if (!$con)
{
die('Could not connect: ' . mysql_error());
}
mysql_select_db("cis_id", $con);
$sql="INSERT INTO nametable (fname, lname)
VALUES
('$$_POST[fname]', '$$_POST[lname]')";
if (!mysql_query($sql,$con))
{
die('Error: ' . mysql_error());
}
echo "1 record added";
mysql_close($con)
?>
</body>
</html>

```

7. Now type the index page URL in our browser and enter some data in to the textboxes. Submit the Query.

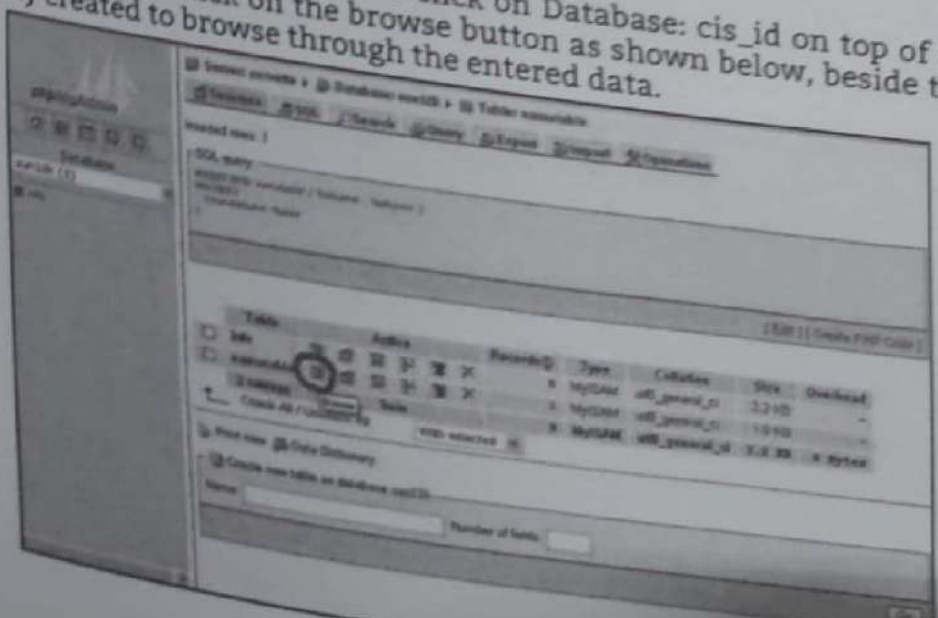
### A small example page to insert some data in to the MySQL database using PHP

Firstname:   
 Lastname:

After Submit Query we will get:

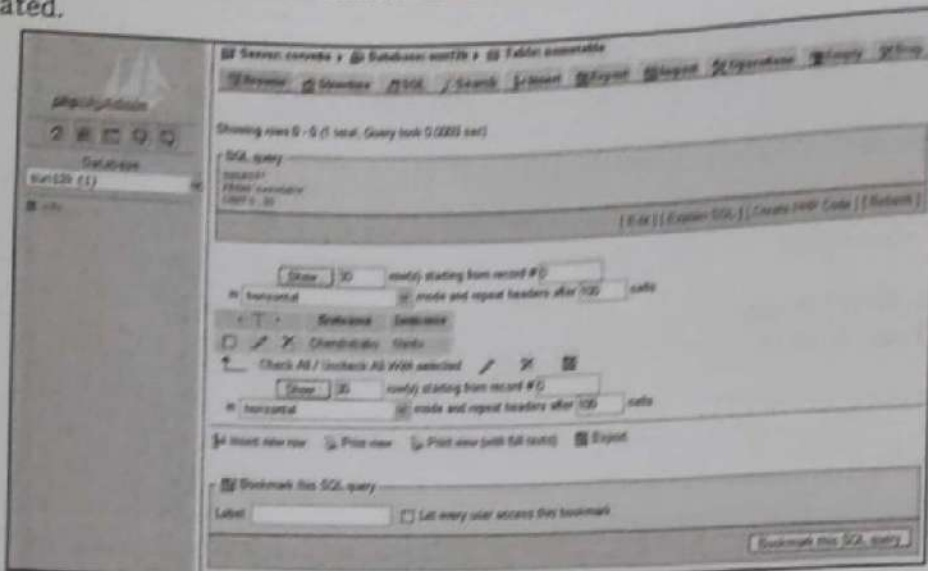
1 record added

8. For browsing the data, we need to click on Database: cis\_id on top of the page to get to the database tables page. Then click on the browse button as shown below, beside the "nametable" which we have already created to browse through the entered data.





9. The following screen then shows us the data being entered in to the MySQL database table being created.



**OUR HIGHLY RECOMMENDED TEXT BOOKS FOR  
THIRD YEAR DIPLOMA COURSES IN ENGINEERING  
AS PER MSBTE'S ADVANCED SYLLABUS 'I' SCHEME  
COMPUTER ENGINEERING AND  
INFORMATION TECHNOLOGY GROUP ; SEMESTER - VI**

- ◆ **PROGRAMING WITH 'PYTHON'**  
Vijay T. Patil, Dr. Meenakshi A. Thalor, Mrs. Jyoti Mante (Khurpade)
- ◆ **MOBILE APPLICATION DEVELOPMENT**  
Mrs. Manisha A. Pokharkar, Prashant D. Somwanshi
- ◆ **WEB BASED APPLICATION DEVELOPMENT USING PHP**  
Prashant D. Somwanshi, Mrs. Mrunal P. Fatangare,  
Mrs. Manisha A. Pokharkar
- ◆ **NETWORK AND INFORMATION SECURITY**  
Mrs. Mrunal P. Fatangare & Others
- ◆ **DATA WAREHOUSING WITH MINING TECHNIQUES**  
Dr. Meenakshi A. Thalor, Prashant D. Somwanshi
- ◆ **WIRELESS AND MOBILE NETWORKS**  
Pratik P. Tewde, Anjum Mujawar, Mrs. Nirmala N. Kamble
- ◆ **CLOUD COMPUTING**  
Dr. Kishor S. Wagh, Mrs. Sharmila K. Wagh

**BOOKS AVAILABLE AT**


**PRAGATI BOOK CENTER** - Email: [pbrpune@pragationline.com](mailto:pbrpune@pragationline.com)


- 157 Budhwar Peth, Opp. Ratan Talkies, Next To Balaji Mandir, Pune 411002 • Mobile : 9657703148
- 676/B Budhwar Peth, Opp. Jogeshwari Mandir, Pune 411002  
Tel : (020) 2448 7459 • Mobile : 9657703147 / 9657703149
- 152 Budhwar Peth, Near Jogeshwari Mandir, Pune 411002  
Mobile : 8087881795
- 28/A Budhwar Peth, Amber Chambers, Appa Balwant Chowk, Pune 411002 • Tel : (020) 6628 1669 • Mobile : 9657703142

**PRAGATI BOOK CORNER** - Email: [niralmumbai@pragationline.com](mailto:niralmumbai@pragationline.com)

- Apurva Building, Shop No. 1, Bhavani Shankar Road, Opp. Shardashram Society, Dadar (W), Mumbai 400028.  
Tel: (022) 2422 3526/6662 5254 • Mobile : 9819935759

[niralipune@pragationline.com](mailto:niralipune@pragationline.com) | [www.pragationline.com](http://www.pragationline.com)

Also find us on  [www.facebook.com/niralibooks](http://www.facebook.com/niralibooks)

 [@nirali.prakashan](https://twitter.com/nirali.prakashan)

**MSBTE'S I SCHEME**

**SALIENT FEATURES**

- This book aims at catering to the needs of Third Year Diploma students as per MSBTE' Revised 'I' Scheme syllabus effective from 2018.
- Complete subject matter has been covered keeping in view the Revised Syllabus.
- Pointwise explanation is given in each topic for ease of study of the relevant chapter.
- Past / Previous Examination Questions have been incorporated in each chapter. This will be extremely useful to students community.
- This book will also be helpful to the students preparing for AMIE (Studentship), Part-time Diploma and Correspondence diploma courses.

**NIRALI PRAKASHAN**  
ADVANCEMENT IN KNOWLEDGE

Abhyudaya Pragati, 1312 Shivaji Nagar,  
Off J.M. Road, Pune 411005.  
Ph. (020) 25512336/7/9  
Fax - (020) 25511 379

