



Yashwantrao
Chavan
Maharashtra
Open University

CMP517

PHP Programming

PHP Programming

Yashwantrao Chavan Maharashtra Open University

Dnyangangotri, Near Gangapur Dam

Nashik-422222

Yashwantrao Chavan Maharashtra Open University

Vice-Chancellor: Prof. E. Vayunandan

SCHOOL OF COMPUTER SCIENCE

Dr. Pramod Khandare Director School of Computer Science Y.C.M.Open University Nashik	Shri. Madhav Palshikar Associate Professor School of Computer Science Y.C.M.Open University Nashik	Dr. P.V. Suresh Director School of Computer and Information Sciences I.G.N.O.U. New Delhi
Dr. Pundlik Ghodke General Manager R&D, Force Motors Ltd. Pune.	Dr. Sahebrao Bagal Principal, Sapkal Engineering College Nashik	Dr. Madhavi Dharankar Associate Professor Department of Educational Technology S.N.D.T. Women's University, Mumbai
Dr. Urmila Shrawankar Associate Professor, Department of Computer Science and Engineering G.H. Rasoni College of Engineering Hingana Road, Nagpur	Dr. Hemant Rajguru Associate Professor, Academic Service Division Y.C.M.Open University Nashik	Shri. Ram Thakar Assistant Professor School Of Continuing Education Y.C.M.Open University Nashik
Mrs. Chetna Kamalskar Assistant Professor School of Science and Technology Y.C.M.Open University, Nashik	Smt. Shubhangi Desle Assistant Professor Student Service Division Y.C.M.Open University Nashik	

Writer/s**Editor****Co-ordinator****Director**

Prof. Parag N. Achaliya
Assistant Professor,
SNJB's Late Sau
K.B. Jain College of
Engineering
Chandwad, Nashik

Prof. Vivek D. Patil
Assistant Professor,
Sandip Institute of
Technology and
Research centre,
Nashik

Ms. Monali R. Borade
Academic Co-ordinator
School of Computer
Science, Y.C.M. Open
University, Nashik

Dr. Pramod Khandare
Director
School of Computer
Science, Y.C.M. Open
University, Nashik

Production

Course Objective

- To understand the fundamentals involved in technologies of Mobile computing
- To introduce Android & understand the basic of Android.
- Design the home screen using UI screen elements.
- Describe the platforms upon which the Android operating system will run.
- To understand android terminologies & resources
- Create an application that uses user interface elements under the Android operating system
- Access and work with databases under the Android operating system
- To share data with another application.

Learning Outcomes:

- Students will be able to understand fundamentals of mobility computing.
- Students will be able to understand working of Android architectures and their applications.
- Students will be able understand the user interface elements and learn the database tools for developing applications on mobile platforms like Android.
- Student will be able to gain the knowledge of deployment of application in actual android device.

Unit No and Name	Title	Counseling Sessions	Weightage
Unit 1: Basics of PHP	❖ Introduction <ul style="list-style-type: none">○ Getting started with PHP ❖ Syntax❖ Echo / Print❖ Variables & Constants❖ Data Types❖ Comments❖ Attributes❖ Operators❖ Decision Making & Loops❖ Predefined Functions❖ Date and Time	4	10
Unit 2: PHP Form Handling	❖ Strings❖ Arrays❖ GET ,POST and REQUEST methods❖ Reading fields from HTML❖ PHP Validations	4	10
Unit 3: File Handling, Session, Cookies in PHP	❖ File Open/Read❖ File Create/Write❖ File Deletion❖ File Upload❖ Cookies❖ Sessions❖ Filters	4	10

Unit 4: Errors and Exception Handling in PHP	<ul style="list-style-type: none"> ❖ Compilation of Errors and Warning <ul style="list-style-type: none"> ○ Parse error ○ syntax error ○ Undefined index ❖ Error Reporting ❖ Exception Handling 	4	10
Unit 5: PHP MySQLi	<ul style="list-style-type: none"> ❖ MySQLi connect ❖ Loop through MySQLi results ❖ Prepared statements in MySQLi ❖ Escaping Strings ❖ Debugging SQL in MySQLi ❖ MySQLi query ❖ How to get data from a prepared statement ❖ MySQLi Insert ID ❖ Close connection ❖ Joins 	4	10
Unit 6: Object Oriented Programming	<ul style="list-style-type: none"> ❖ Defining PHP classes ❖ Creating objects in PHP ❖ Calling Member Functions. ❖ Constructor functions ❖ Destructor ❖ Inheritance ❖ Function Overriding ❖ Access Specifiers ❖ Interfaces ❖ Abstract Classes ❖ Static and Final Keywords ❖ Calling Parent Constructors ❖ Namespaces ❖ Functions 	3	10
Unit 7: PHP Frameworks and Laravel	<ul style="list-style-type: none"> ❖ Introduction to Framework <ul style="list-style-type: none"> ○ Laravel ○ CodeIgniter ○ CakePHP ❖ Yii ❖ MVC(Model View controller) Introduction ❖ Laravel Installation ❖ Laravel Database Connectivity 	4	10
Unit 8: Content Management system and WordPress	<ul style="list-style-type: none"> ❖ Introduction to CMS <ul style="list-style-type: none"> ○ WordPress ○ Joomla ○ Drupal ○ Magento ❖ WordPress <ul style="list-style-type: none"> ○ Home ○ Overview ○ Installation ○ Dashboard 	3	10

	<ul style="list-style-type: none"> ○ Add, Delete, Publish Post ○ Media Library ○ Add, Delete, Publish Page 		
		30	80

Reference Books:

1. **Php: The Complete Reference** by steven holzner, Publisher: Mcgraw Hill, Latest edition
2. **Learning PHP, MySQL & JavaScript:** With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5) by Robin Nixon, O'Reilly Media, 5th edition (2018).
3. The Joy of PHP Programming: A Beginner's Guide to Programming Interactive Web Applications with PHP and MySQL, Author –Alan Forbes, Latest Edition – Fifth Edition, Publisher – Plum Island Publishing LLC.
4. Code Smart: The Laravel Framework Version 5 for Beginners by Dayle Rees.
5. Building Web Apps with WordPress: WordPress As An Application Framework by Brian Messenlehner and Jason Coleman Foreword by Brad Williams.
6. PHP Cookbook: Solutions & Examples For PHP Programmers by Adam Trachtenberg and David Sklar

Note: This Study material is still under development/editing process and is being made available for the sole purpose of reference. Final edited copies will be made available once ready.

Chapter 1. BASICS OF PHP

Learning Objectives:

After successful completion of this unit, you will be able to

1. Understand origin of PHP
2. Understand basics of PHP.
3. Summarize variables, constants and data types.
4. Summarize conditional statement in PHP.
5. Summarize built in functions in PHP.

This chapter will introduce you to PHP. You will learn how it came about, what it looks like, and why it is the best server-side technology. You will also be exposed to the most important features of the language.

This chapter will let you poke around the different variables & constants, data types, operators, decision making & loops, predefined functions in PHP.

1.1 Introduction

The term PHP is an acronym for **PHP: Hypertext Preprocessor**. PHP is a server-side scripting language designed specifically for web development. PHP can be easily embedded in HTML files and HTML codes can also be written in a PHP file. The thing that differentiates PHP with client-side language like HTML is, PHP codes are executed on the server whereas HTML codes are directly rendered on the browser. In our entire course we will be studying PHP 7 version and on an Ubuntu OS.

1.2 The Origins of PHP

Wonderful things come from singular inspiration. PHP began life as a simple way to track visitors to RasmusLerdorf's online resume. It also could embed SQL queries in Web pages. But as often happens on the Web, admirers quickly asked for their own copies. As a proponent of the Internet's ethic of sharing, as well as a generally agreeable person, Rasmus unleashed upon an unsuspecting Web his Personal Home Page Tools version 1.0.

"Unleashed upon himself" may be more accurate. PHP became very popular. A consequence was a flood of suggestions. PHP 1.0 filtered input, replacing simple commands for HTML. As its popularity grew, people wondered if it couldn't do more.

Loops, conditionals, rich data structures—all the conveniences of modern structured programming seemed like a next logical step. Rasmus studied language parsers, read about YACC and GNU Bison, and created PHP 2.0.

PHP 2.0 allowed developers to embed structured code inside HTML tags. PHP scripts could parse data submitted by HTML forms, communicate with databases, and make complex calculations on the fly. And it was very fast, because the freely available source code compiled into the Apache Web server. A PHP script executed as part of the Web server process and required no forking, often a criticism of Common Gateway Interface (CGI) scripts.

PHP was a legitimate development solution and began to be used for commercial Websites. In 1996 Clear Ink created the SuperCuts site ([www. supercuts.com](http://www.supercuts.com)) and used PHP to create a custom experience for the Web surfer.

1.3 PHP Installation

LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP.

Following are the steps for installation:

Step 1: Prerequisites

You must have sudo privileges account access to the Linux system. Login to your system and upgrade the current packages to the latest available version.

```
sudo apt update
```

```
sudo apt upgrade
```

Also, install the below packages on your system.

```
sudo apt install ca-certificates apt-transport-https
```

Step 2: Install Apache Web Server

In this section, we will install a web server on your VPS. We can install Apache, or nginx as a web server. For the purpose, we will install the Apache web server. Apache is a fast and secure web server and one of the most popular and widely used web server in the world. To install the Apache web server, run the following command on your server:

```
apt-get install apache2
```

After the installation is complete, you should start Apache:

```
systemctl start apache2
```

Also, you can enable Apache to start automatically on server boot:

```
systemctl enable apache2
```

To check the status of the Apache web server and make sure it is up and running, you can use the following command:

```
systemctl status apache2
```

To verify that Apache is running, you can also open your web browser and enter your server IP address, (e.g. http://your_server_ip_address). If Apache is successfully installed, you should see the Apache default welcome page.

Step 3: Install MySQL

MySQL is a powerful database management system used for organizing and retrieving data. To install MySQL, open terminal and type in these commands:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

During the installation, MySQL will ask you to set a root password. If you miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell.

Once you have installed MySQL, we should activate it with this command:

```
sudo mysql_install_db
```

Finish up by running the MySQL set up script:

```
sudo /usr/bin/mysql_secure_installation
```

The prompt will ask you for your current root password.

Type it in.

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Then the prompt will ask you if you want to change the root password. Go ahead and choose N and move on to the next steps.
It's easiest just to say Yes to all the options. At the end, MySQL will reload and implement the new changes.

```
By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MySQL comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...
```

Step 4: Install PHP7

PHP is an open source web scripting language that is widely used to build dynamic webpages.

To install PHP, open terminal and type in this command.

```
sudo apt-get install -y php7.0
```


1.4 PHP Basic Syntax

PHP or Hypertext Preprocessor is a widely used open-source general purpose scripting language and can be embedded with HTML. PHP scripts can be written anywhere in the document within PHP tags along with normal HTML.

1.4.1 Writing PHP Statements

A PHP statement tells PHP to perform an action. One of the most common PHP statements is the echo statement. Its purpose is to display output. For instance, take a look at the following echo statement:

```
echo "Hi";
```

An echo statement says to output everything that is between the double quotes (""). So, this statement tells PHP to output the word Hi.

The echo statement is a *simple statement*. PHP simple statements end with a semicolon (;). PHP reads a simple statement until it encounters a semicolon (or the PHP closing tag). PHP ignores white space. It doesn't care how many lines it reads. It doesn't consider the content or the syntax of the statement. It just reads until it finds a semicolon and then interprets the entire content as a single statement.

Another common PHP statement similar to echo statement is print, which is also used to output data to the screen. The print statement can be used with or without parentheses: print or print().

The following example shows how to output text with the print

```
print "Hello world!";  
print "I'm about to learn PHP!";
```

Leaving out the semicolon is a common error, resulting in an error message that looks something like this:

Parse error: expecting `,' or `;' in **file.php** on line 6

Notice that the error message gives you the line number where it encountered problems. Usually, the error is that the semicolon was left off in the line before the indicated line. In this case, the semicolon is probably missing on line 5.

You may prefer to use an editor that displays line numbers. Debugging your PHP scripts is much easier this way. Otherwise, you need to count the lines from the top of the script to find the line containing the error. If your script contains six lines, counting them is no big deal. If your script contains 553 lines, however, this is less than fun. Some editors allow you to indicate a line number, and the editor takes you directly there. As far as PHP is concerned, an entire script full of simple statements can be written in one long line, as long as the statements are separated by semicolons.

However, a human would have a tough time reading such a script. Therefore, you should put simple statements on separate lines.

Sometimes several statements are combined into a block, which is enclosed by curly braces ({}). Statements in a block execute together. A common use of a block is in a conditional statement where statements are executed only if certain conditions are met. For instance, you may want to include the following instructions:

```
if (time == midnight)  
{  
    put on pajamas;  
    brush teeth;  
    go to bed;  
}
```

The statements are enclosed in curly braces to ensure they execute as a block. If it's midnight, then all three actions within the block are performed. If the time is not midnight, none of the statements execute (no pajamas, no clean teeth; no going to bed).

PHP statements that use blocks, such as if statements, are called *complex statements*. PHP reads the entire complex statement, not stopping at the first semicolon it encounters. PHP knows to expect one or more blocks and looks for the ending curly brace of the last block in complex statements. Notice that there is a semicolon before the ending brace. This semicolon is required, but no semicolon is required after the ending curly brace.

Notice that the statements inside the block are indented. Indenting is not necessary for PHP. Indenting is strictly for readability. You should indent the statements in a block so that people reading the script can tell more easily where a block begins and ends. One of the more common mistakes when writing scripts is to leave out a closing curly brace, particularly when writing blocks inside blocks inside blocks. Tracking down a missing brace is much easier when the blocks are indented.

1.4.2 Building Scripts

To build a script, you add PHP statements one after another to a file that you name with a .php extension. Actually, if you are wise, you write the script on paper first, unless the script is very simple or you are quite experienced.

Planning makes programming much less prone to errors. If you're writing a PHP script for your Web site, you insert the PHP statements into the file that contains the HTML for your Web page. If you're writing a script that will run independent of the Web, you type the PHP statements into a file and then you run the script by calling PHP directly.

1.5 PHP Tags or Escaping To PHP

The mechanism of separating a normal HTML from PHP code is called the mechanism of Escaping To PHP. There are various ways in which this can be done. Few methods are already set by default but in order to use few others like Short-open or ASP-style tags we need to change the configuration of php.ini file. These tags are also used for embedding PHP within HTML. There are 4 such tags available for this purpose:-

1. **Canonical PHP Tags:** The script starts with `<?php` and ends with `?>`. These tags are also called 'Canonical PHP tags'. Every PHP command ends with a semi-colon (;). Let's look at the *hello world* program in PHP:

```
<?php
# Here echo command is used to print
echo "Hello, world!";
?>
```

Output:
Hello, world!

2. **SGML or Short HTML Tags:** These are the shortest option to initialize a PHP code. The script starts with `<?` and ends with `?>`. This will only work by setting the `short_open_tag` setting in php.ini file to 'on'.

Example:

```
<?
# Here echo command will only work if
# setting is done as said before
echo "Hello, world!";
?>
```

3. **HTML Script Tags:** These are implemented using script tags. This syntax is removed in PHP 7.0.0 so its no more used.

Example:

```
<script language="php">
echo "hello world!";
</script>
```

4. **ASP Style Tags:** To use this we need to set the configuration of php.ini file. These are used by Active Server Pages to describe code blocks. These starts with <% and ends with %>.

Example:

```
<%
# Can only be written if setting is turned on
# to allow %
echo "hello world";
%>
```

1.6Running a PHP script:

For Linux/ Unix, if you have a file named testcli.php containing this PHP code, you can run it from the command line by having the file in the same directory where PHP is installed and by typing the following:

```
php testcli.php
```

Or you can type the entire path name to PHP, as in the following example:

```
/usr/local/php/cli/php testcli.php
```

The CLI version of PHP differs from the CGI version in the following ways:

1. **Outputting HTTP headers:** Because the CGI version sends its output to the Web server and then to the browser, it outputs the HTTP *headers* (statements the Web server and browser use to communicate with each other). Thus, the following is the output when the CGI version runs the script in the previous example:

```
Content-type: text/html
```

```
X-Powered-By: PHP/7.0
```

```
This line brought to you by PHP
```

You don't see the two headers on your Web page, but PHP for the Web sends these headers because the Web server needs them. The CLI version, on the other hand, does not automatically send the HTTP headers because it is not sending its output to a Web server. The CLI output is limited to the following:

This line brought to you by PHP

2. **Formatting error messages:** The CGI version formats error messages with HTML tags, because the errors are expected to be received by a browser. The CLI version does not use HTML formatting for error messages; it outputs error messages in plain text.
3. **Providing argc and argv by default:** The argc and argv variables allow you to supply data to the script from the command line (similar to argc and argv in C and other languages). You aren't likely to want to pass data to the CGI version, but you are likely to want to pass data to the CLI version. Therefore, argv and argc are available by default in the CLI version and not in the CGI version.

When you run PHP CLI from the command line, you can use several options that affect the way PHP behaves. For instance, -v is an option that displays the version of PHP being accessed. To use this option, you would type the following:

```
php -v
```

Table 1.1 shows the most useful PHP command-line options.

Option	What it Does
-c	Defines the path to the php.ini file to be used. This can be a different php.ini file than the one used by the CGI version. For example, -c /usr/local/php/cli/php.ini.
-f	Identifies the script to be run. For example, php -f /myfiles/testcgi.php.
-h	Displays a help file.
-i	Displays PHP information in text output. Gives the same information as phpinfo()
-l	Checks the script file for errors, but doesn't actually execute the code
-m	Lists the modules that are compiled into PHP
-r	Runs PHP code entered at the command line. For example, php -r 'print('Hi');'.
-v	Displays the version number of PHP.

1.7 Variables in PHP

Variables are containers that hold information. First, you give a variable a name, and then you can store information in it. For example, you could name a variable \$age and store the number 21 in it. After you store information in a variable, you can use that variable later in the script. When using PHP on the Web, variables are often used to store the information that users type into an HTML form, such as their names. You can then use the variable later in the script, perhaps to personalize a Web page by displaying the user's name, as in, for example, Welcome Sam Smith.

In this section, you find out how to create variables, name them, and store information in them. You also discover how to handle errors.

1.7.1 Naming Variables

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

The rules for variable names are as follows:

1. **All variable names start with a dollar sign (\$).** This tells PHP that it is a variable name.
2. **Variable names can be any length.**

3. **Variable names can include letters, numbers, and underscores only.**
4. **Variable names must begin with a letter or an underscore.** They cannot begin with a number.
5. **Uppercase and lowercase letters are not the same:** \$favoritecity and \$Favoritecity are not the same variable. If you store information in \$FavoriteCity, you can't retrieve that information later in the script by using the variable name \$favoriteCity.

The following are valid variable names:

```
$_name
$first_name
$name3
$name_3
```

The following variable names cause error messages:

```
$3name
$name?
$first+name
$first.name
```

1.7.2 Creating variables

Storing information in a variable creates it. To store information in a variable, you use a single equal sign (=).

For example,

The following four PHP statements assign information to variables:

```
$age = 21;
$price = 20.52;
$temperature = -5;
$name = "Ramesh";
```

In these examples, notice that the numbers are not enclosed in quotes, but the name, which is a string of characters, is. The quotes tell PHP that the characters are a string, handled by PHP as a unit. Without the quotes, PHP doesn't know the characters are a string and won't handle them correctly.

Whenever you put information into a variable that did not previously exist, you create that variable.

For example, suppose you use the following PHP statements at the top of your script:

```
$color = "blue";
$color = "red";
```

If the first statement is the first time you mention the variable \$color, this statement creates the variable and sets it to "blue". The next statement changes the value of \$color to "red".

You can store the value of one variable in another variable, as shown in the following statements:

```
$name1 = "Suresh";
$name2 = "Patil";
$favorite_name = $name2;
```

After these statements are executed, the variable \$favorite_name contains the value "Suresh".

You can create a variable without storing any information in it.

For example, the following statement creates a variable:

```
$city = "";
```

1.7.3 Displaying variable values

The quickest way to display the value stored in a variable is with the `print_r` statement. You can output the value of a variable as in the following statements:

```
$today = "Sunday";  
print_r($today);
```

The output from the preceding statements is Sunday.

You can also display the value by using an `echo` statement. If you used the following PHP statements

```
$age = 21;  
echo $age;
```

in a PHP section, the output would be 21.

Using an `echo` statement of the preceding form, with one variable only, provides the same basic output as the `print_r` statement. However, you can do a lot more with the `echo` statement. You can output several items and include numbers and strings together.

For example, suppose the variable `$name` has the value Ramesh. You can include the following line in an HTML file:

```
<p>Welcome <?php echo $name ?></p>
```

The output on the Web page is as follows:

Welcome Ramesh

If you use a variable that does not exist, you get a warning message. For example, suppose you intend to display `$age`, but type the following statement by mistake:

```
echo $aeg;
```

You get a notice that looks like the following:

```
Notice: Undefined variable: aeg in c:\testvar.php on line 5
```

The notice points out that you're using a variable that has not yet been given a value.

1.8 Working with Constants

Constants are similar to variables. Constants are given names, and values are stored in them. However, constants are constant; they can't be changed by the script. After you set the value for a constant, it stays the same. If you use a constant for weather and set it to sunny, it can't be changed.

1.8.1 Creating constants

Constants are set by using the `define` statement. The general format is as follows:

```
define("constantname", "constantvalue");
```

For example, to set a constant with the weather, use the following statement:

```
define("WEATHER", "Sunny");
```

This statement creates a constant called `WEATHER` and sets its value to "Sunny".

When naming constants, use descriptive names, as you do for variables. However, unlike variables, constant names do not begin with a dollar sign (\$). By convention, constants are given names that are all uppercase so you can see easily that they're constants. However, PHP accepts lowercase letters without complaint. You can store either a string or a number in a

constant. The following statement, which defines a constant named INTEREST and assigns to it the value .01, is perfectly okay with PHP:

```
define ("INTEREST",.01);
```

Constants should not be given names that are keywords for PHP. *Keywords* are words that have meaning for PHP, such as echo, and they can't be used as constants because PHP treats them as the PHP feature of the same name. PHP will let you define a constant ECHO without giving an error message, but it will have a problem when you try to use the constant. For example, if you use the following statement:

```
echo ECHO;
```

PHP gets confused and displays an error message. It sees the constant as the beginning of another echo statement, but it can't find all the things it needs to complete the first echo statement.

Some PHP keywords include the following:

And	as	break
Case	class	const
Continue	declare	default
Die	do	echo
Else	empty	eval
Exit	for	foreach
Function	global	if
Include	list	new
Or	print	require
Return	switch	use
var	while	

If you're baffled by some code that looks perfectly okay but refuses to work correctly, even after numerous changes, try changing the name of a constant. It's possible that you are using an obscure keyword for your constant, and that's causing your problem. This doesn't happen often, but it's possible. Although you can use keywords for variable names, because the beginning \$ tells PHP the keyword is a variable name, you probably shouldn't. It causes too much confusion for the humans involved.

1.8.2 Understanding when to use constants

If you know the value of something won't change during the script, use a constant. Using a constant allows you to use a descriptive name, making the script clearer. For example, PRODUCT_COST is much clearer than 20.50. Using a constant allows you to set the value once at the beginning of the script. If this value ever needs to be changed, using constants allows you to change it in only one place, instead of finding and changing the value in 20 different places throughout the script. One change is better than 20. It's less work and lessens the likelihood of missing a place that needed to be changed, leading to unknown and unseen havoc. Using a constant ensures that the value won't be changed accidentally somewhere in the script, leading to the wrong value being used in statements later in the script.

Suppose you have a script that must change money from one currency to another by multiplying the dollar amount by the exchange rate. For example, if the exchange rate from Indian Rupee to U.S. dollars is 0.014, you can write the following code:

```
<?php
$Indian_Rupee = 20.00;
$US_dollars = $Indian_Rupee * 0.014;
?>
```

Now, suppose your script contains 40,000 lines of code and you need to convert Indian Rupee to U.S. dollars in 50 different places in the script. So you use the preceding code in 50 different places. Then you realize that the exchange rate is likely to change every week, so you would need to go through this script every week and change 0.014 to something else, manually, in 50 different places.

That's a lot of work.

A better way to handle this is to put the exchange rate in a variable so you could change it only in one place. You change your script to the following:

```
<?php
$rate = 0.014;
$Indian_Rupee = 20.00;
$US_dollars = $Indian_Rupee * $rate;
?>
```

You set \$rate at the beginning of the script. Then you can use the two lines that convert the currency in all 50 parts of the script. This is clearly a better option. When the rate changes, you need to change the rate in only one place. For example, if the exchange rate changes to 0.20 next week, you just change the first line of the script to the following:

```
$rate = 0.20;
```

This would work. However, \$rate is not a very descriptive name. Remember that your script is 40,000 lines of code and the 2 lines of code that convert currency are used in 50 different places. Suppose somewhere in the middle of your script you need to add some code to compute interest. Suppose you accidentally use the following code somewhere in the middle of your script:

```
$interest_rate = 20;
$rate = $interest_rate-1;
$amount = $principal * $rate;
```

All the places after this code will have a different value for rate; the 0.014 that you set at the beginning of your script will be replaced by the 19 set by this code. You can guard against this by using more descriptive variable names.

Or an even better option is to use a constant, as in the following script:

```
<?php
define("RATE",0.014);
$Indian_Rupee = 20.00;
$US_dollars = $Indian_Rupee * $RATE;
?>
```

Now you are using a constant, RATE, that can't be changed in the script. If you try to add the line

RATE = 20; in the middle of your script, PHP won't allow it. So, you won't make the mistake that you made with the variable.

Next week when the exchange rate changes to 1.53, you just edit your script as follows:

```
<?php
define("RATE",0.014);
$Indian_Rupee = 20.00;
$US_dollars = $Indian_Rupee * $RATE;
?>
```

Of course, this would be even better if you used a more descriptive name, such as the following:

```
define("IR_TO_US",0.014);
```


Keep in mind that mistakes that seem impossible to make when you're looking at a ten-line script, become entirely possible when you think in terms of scripts with thousands of lines of code, especially scripts with more than one programmer involved.

If you know the value of something won't change during the script, use a constant. If you need to manipulate the value somewhere in the script, use a variable.

1.8.3 Displaying constants

You can determine the value of a constant by using `print_r` as follows:

```
print_r(IR_TO_US);
```

You can also use a constant in an `echo` statement:

```
echo IR_TO_US;
```

When you `echo` a constant, you can't enclose it in quotes. If you do, it `echoes` the constant name rather than the value. You can `echo` the constant as shown in the preceding example, or you can enclose it with parentheses. You can build more complicated output statements by using commas, as in the following example:

```
echo "The exchange rate is $", IR_TO_US;
```

The output from this statement is the following:

```
The exchange rate is $0.014.
```

Notice that the dollar sign is inside the quoted string in the first output string, not in the second output item as part of the constant name.

1.8.4 Utilizing built-in PHP constants

PHP has many built-in constants that you can use in your scripts. For example, the constant `__LINE__` has a value that is the line number where it is used, and `__FILE__` contains the name of the file in which it is used. (These constants begin with two underscores and end with two underscores.) For example, you can use the following statement:

```
echo __FILE__;
```

The output looks similar to the following:

```
c:\program files\apache group\apache\htdocs\testvar2.php
```

PHP has many other built-in constants. For example, `E_ALL` and `E_ERROR` are constants you can use to affect how PHP handles errors.

1.9 Data Types in PHP

Data Types defines the type of data a variable can store. PHP allows eight different types of data types. All of them are discussed below. The first five are called simple data types and the last three are compound data types:

- 1.9.1 Integer:** Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point. They can be decimal (base 10), octal (base 8) or hexadecimal (base 16). The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The range of integers must lie between -2^{31} to 2^{31} .

Example:

```
<?php
// decimal base integers
```

```

$dec1= 50;
$dec2= 654;
// octal base integers
$octal1= 07;
// hexadecimal base integers
$octal= 0x45;
$sum= $dec1+ $dec2;
echo$sum;
?>

```

Output:
704

- 1.9.2 Double:** Can hold numbers containing fractional or decimal part including positive and negative numbers. By default, the variables add a minimum number of decimal places. Example:

```

<?php
$val1= 50.85;
$val2= 654.26;
$sum= $val1+ $val2;
echo$sum;
?>

```

Output:
705.11

- 1.9.3 String:** Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes but it will be treated differently while printing variables. Example:

```

<?php
$name= "Krishna";
echo"The name of the Geek is $name \n";
echo"The name of the geek is $name';
?>

```

Output:
The name of the Geek is Krishna
The name of the geek is \$name

- 1.9.4 NULL:** These are special types of variables that can hold only one value i.e., NULL. We follow the convention of writing it in capital form, but its case sensitive.

Example:

```

<?php
$nm= NULL;
echo$nm; // This will give no output
?>

```

- 1.9.5 Boolean:** Hold only two values, either TRUE or FALSE. Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean. Apart from NULL, 0 is also considering as false in boolean. If a string is empty then it is also considered as false in boolean data type.

Example:

```
<?php
    if(TRUE)
        echo"This condition is TRUE";
    if(FALSE)
        echo"This condition is not TRUE";
?>
```

Output:

This condition is TRUE

- 1.9.6 Arrays:** Array is a compound data-type which can store multiple values of same data type. Below is an example of array of integers.

```
<?php
$array= array( 10, 20 , 30);
echo"First Element: $array[0]\n";
echo"Second Element: $array[1]\n";
echo"Third Element: $array[2]\n";
?>
```

Output:

First Element: 10

Second Element: 20

Third Element: 30

- 1.9.7 Objects:** Objects are defined as instances of user defined classes that can hold both values and functions. This is an advanced topic and will be discussed in details in further articles.

- 1.9.8 Resources:** Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource.

1.10 PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators
7. Array operators

1.10.1 PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$\$x + \y	Sum of $\$x$ and $\$y$
-	Subtraction	$\$x - \y	Difference of $\$x$ and $\$y$
*	Multiplication	$\$x * \y	Product of $\$x$ and $\$y$
/	Division	$\$x / \y	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \y	Remainder of $\$x$ divided by $\$y$
**	Exponentiation	$\$x ** \y	Result of raising $\$x$ to the $\$y$ 'th power (Introduced in PHP 5.6)

1.10.2 PHP Assignment Operators

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

Assignment	Same as...	Description
$x = y$	$x = y$	The left operand gets set to the value of the expression on the right
$x += y$	$x = x + y$	Addition
$x -= y$	$x = x - y$	Subtraction
$x *= y$	$x = x * y$	Multiplication
$x /= y$	$x = x / y$	Division
$x \% = y$	$x = x \% y$	Modulus

1.10.3 PHP Comparison Operators

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

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

1.10.4 PHP Increment / Decrement Operators:

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

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

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

1.10.5 PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

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

1.10.6 PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

1.10.7 PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	\$x + \$y	Union of \$x and \$y
==	Equality	\$x == \$y	Returns true if \$x and \$y have the same key/value pairs
===	Identity	\$x === \$y	Returns true if \$x and \$y have the same key/value pairs in the same order and of the same types

!=	Inequality	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Inequality	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Non-identity	\$x !== \$y	Returns true if \$x is not identical to \$y

1.11 Using Conditional Statements:

A *conditional statement* executes a block of statements only when certain conditions are true. Here are two useful types of conditional statements:

1.11.1 if statement: Sets up a condition and tests it. If the condition is true, a block of statements is executed.

An if statement tests conditions, executing a block of statements when a condition is true. The general format of an if conditional statement is as follows:

```

if ( condition )
{
    block of statements
}
elseif ( condition )
{
    block of statements
}
else
{
    block of statements
}

```

The if statement consists of three sections:

i. if: This section is required. It tests a condition: **If the condition is true:** The block of statements is executed. After the statements are executed, the script moves to the next instruction following the conditional statement; if the conditional statement contains any elseif or else sections, the script skips over them.

If the condition is not true: The block of statements is not executed.

ii. elseif: This section is optional. You can use more than one elseif section if you want. It also tests a condition: **If the condition is true:** The block of statements is executed. After executing the block of statements, the script goes to the next instruction following the conditional statement; if the if statement contains any additional elseif sections or an else section, the script skips over them.

If the condition is not true: The block of statements is not executed. The script skips to next instruction, which can be an elseif, an else, or the next instruction after the if conditional statement.

iii. Else: This section is also optional. Only one else section is allowed. This section does not test a condition, rather it executes the block of statements. If the script has entered this section, it means that the if section and all the elseif sections are not true.

Here's an example. Pretend you're a teacher. The following if statement, when given a test score, sends your student a grade and a snappy little text message. It uses all three sections of the if statement, as follows:

```
if ($score > 92 )
{
$grade = "A";
$message = "Excellent!";
}
elseif ($score <= 92 and $score > 83 )
{
$grade = "B";
$message = "Good!";
}
elseif ($score <= 83 and $score > 74 )
{
$grade = "C";
$message = "Okay";
}
elseif ($score <= 74 and $score > 62 )
{
$grade = "D";
$message = "Uh oh!";
}
else
{
$grade = "F";
$message = "Doom is upon you!";
}
echo $message."\n";
echo "Your grade is $grade\n";
```

The if conditional statement proceeds as follows:

- a) The value in \$score is compared to 92. If \$score is greater than 92, \$grade is set to A, \$message is set to Excellent!, and the script skips to the echo statement. If \$score is 92 or less, \$grade and \$message are *not* set, and the script skips to the elseif section.
- b) The value in \$score is compared to 92 and to 83. If \$score is 92 or less *and* greater than 83, \$grade and \$message are set, and the script skips to the echo statement. If \$score is 83 or less, \$grade and \$message are *not* set, and the script skips to the second elseif section.
- c) The value in \$score is compared to 83 and to 74. If \$score is 83 or less *and* greater than 74, \$grade and \$message are set, and the script skips to the echo statement. If \$score is 74 or less, \$grade and \$message are *not* set, and the script skips to the next elseif section.
- d) The value in \$score is compared to 74 and to 62. If \$score is 74 or less *and* greater than 62, \$grade and \$message are set, and the script skips to the echo statement. If \$score is 62 or less, \$grade and \$message are *not* set, and the script skips to the else section.
- e) \$grade is set to F, and \$message is set to Doom is upon you!.
The script continues to the echo statement.

When the block to be executed by any section of the if conditional statement contains only one statement, the curly braces are not needed. For example, say the preceding example had only one statement in the blocks, as follows:

```
if ($grade > 92 )
{
$grade = "A";
}
```

You could write it as follows:

```
if ($grade > 92 )
$grade = "A";
```

This shortcut can save some typing, but when several if statements are used, it can lead to confusion.

Nesting if statements

You can have an if conditional statement inside another if conditional statement. Putting one statement inside another is called *nesting*. For example, suppose you need to contact all your customers who live in Idaho. You plan to send e-mail to those who have e-mail addresses and send letters to those who do not have e-mail addresses. You can identify the groups of customers by using the following nested if statements:

```
if ( $custState == "ID" )
{
if ( $EmailAdd = "" )
{
$contactMethod = "letter";
}
else
{
$contactMethod = "email";
}
}
else
{
$contactMethod = "none needed";
}
```

These statements first check to see if the customer lives in Idaho. If the customer does live in Idaho, the script tests for an e-mail address. If the e-mail address is blank, the contact method is set to letter. If the e-mail address is not blank, the contact method is email. If the customer does not live in Idaho, the else section sets the contact method to indicate that the customer will not be contacted at all.

1.11.2 Switch statement: Sets up a list of alternative conditions. It tests for the true condition and executes the appropriate block of statements.

For most situations, the if conditional statement works best. However, sometimes you have a list of conditions and want to execute different statements for each condition. For example, suppose your script computes sales tax. How do you handle the different state sales tax rates? The switch statement was designed for such situations.

The switch statement tests the value of one variable and executes the block of statements for the matching value of the variable. The general format is as follows:

```
switch ( $variablename )
{
```

```

casevalue :
block of statements;
break;
casevalue :
block of statements;
break;
...
default:
block of statements;
break;
}

```

The switch statement tests the value of *\$variablename*. The script then skips to the case section for that value and executes statements until it reaches a break statement or the end of the switch statement. If there is no case section for the value of *\$variablename*, the script executes the default section. You can use as many case sections as you need. The default section is optional. If you use a default section, it's customary to put the default section at the end, but as far as PHP is concerned, it can go anywhere. The following statements set the sales tax rate for different states:

```

switch ( $custState )
{
case "OR" :
$salestaxrate = 0;
break;
case "CA" :
$salestaxrate = 1.0;
break;
default:
$salestaxrate = .5;
break;
}
$salestax = $orderTotalCost * $salestaxrate;

```

In this case, the tax rate for Oregon is 0, the tax rate for California is 100 percent, and the tax rate for all the other states is 50 percent. The switch statement looks at the value of *\$custState* and skips to the section that matches the value. For example, if *\$custState* is TX, the script executes the default section and sets *\$salestaxrate* to .5. After the switch statement, the script computes *\$salestax* at .5 times the cost of the order. The break statements are essential in the case section. If a case section does not include a break statement, the script does *not* stop executing at the end of the case section. The script continues executing statements past the end of the case section, on to the next case section, and continues until it reaches a break statement or the end of the switch statement. This is a problem for every case section except the last one because it will execute sections following the appropriate section. The last case section in a switch statement doesn't actually require a break statement. You can leave it out. However, it's a good idea to include it for clarity and consistency.

1.11.3 Repeating Actions by Using Loops

Loops are used frequently in scripts to set up a block of statements that repeat. The loop can repeat a specified number of times. For example, a loop that echoes all the state capitals needs to repeat 50 times. Or the loop can repeat until a certain condition is met. For example, a loop that echoes the names of all the files in a directory needs to repeat

until it runs out of files, regardless of how many files there are. Here are three types of loops:

1. **A for loop:** Sets up a counter; repeats a block of statements until the counter reaches a specified number. The most basic for loops are based on a counter. You set the beginning value for the counter, set the ending value, and set how the counter is incremented each time the statement block is executed. The general format is as follows:

```
for (startingvalue;endingcondition;increment)
{
    block of statements;
}
```

Within the for statement, you need to fill in the following values:

- i. *startingvalue*: The *startingvalue* is a statement that sets up a variable to be your counter and sets it to your starting value. For example, the statement `$i=1`; sets `$i` as the counter variable and sets it equal to 1. Frequently, the counter variable is started at 0 or 1. The starting value can be a number, a combination of numbers (like `2 + 2`), or a variable.
- ii. *endingcondition*: The *endingcondition* is a statement that sets your ending value. As long as this statement is true, the block of statements keeps repeating. When this statement is not true, the loop ends. For example, the statement `$i<10`; sets the ending value for the loop to 10. When `$i` is equal to 10, the statement is no longer true (because `$i` is no longer less than 10), and the loop stops repeating. The statement can include variables, such as `$i<$size`;
- iii. *increment*: A statement that increments your counter. For example, the statement `$i++`; adds 1 to your counter at the end of each block of statements. You can use other increment statements, such as `$i+=1`; or `$i--`;. A basic for loop sets up a variable, like `$i`, that is used as a counter. This variable has a value that changes during each loop. The variable `$i` can be used in the block of statements that is repeating.

For example, the following simple loop displays Hello World! three times:

```
for ($i=1;$i<=3;$i++)
{
    echo "$i. Hello World!<br>";
}
```

The statements in the block do not need to be indented. PHP doesn't care whether they're indented. However, indenting the blocks makes it much easier for you to understand the script.

You can nest for loops inside of for loops. Suppose you want to print out the times tables from 1 to 9. You can use the following statements:

```
for($i=1;$i<=9;$i++)
{
    echo "\nMultiply by $i \n";
    for($j=1;$j<=9;$j++)
    {
        $result = $i * $j;
        echo "$i x $j = $result\n";
    }
}
```

The output is as follows:

Multiply by 1

$1 \times 1 = 1$
 $1 \times 2 = 2$
 ...
 $1 \times 8 = 8$
 $1 \times 9 = 9$
 Multiply by 2
 $2 \times 1 = 2$
 $2 \times 2 = 4$
 ...
 $2 \times 8 = 16$
 $2 \times 9 = 18$
 Multiply by 3
 $3 \times 1 = 3$
 and so on.

1.11.4 A while loop: Sets up a condition; checks the condition, and if it's true, repeats a block of statements until the condition becomes false.

A while loop continues repeating as long as certain conditions are true. The loop works as follows:

1. You set up a condition.
2. The condition is tested at the top of each loop.
3. If the condition is true, the loop repeats. If the condition is not true, the loop stops.

The following is the general format of a while loop:

```

while ( condition )
{
  block of statements
}
  
```

The following statements set up a while loop that looks through an array for an apple:

```

$fruit = array ("orange", "apple", "grape");
$testvar = "no";
$k = 0;
while ( $testvar != "yes" )
{
  if ($fruit[$k] == "apple" )
  {
    $testvar = "yes";
    echo "apple\n";
  }
  else
  {
    echo "$fruit[$k] is not an apple\n";
  }
  $k++;
}
  
```

These statements generate the following output:

```

orange is not an apple
apple
  
```

It's possible to write a while loop that is infinite — that is, a loop that loops forever. You can easily, without intending to, write a loop in which the condition is always true. If the condition never becomes false, the loop never ends.

1.11.5 A do..while loop: Sets up a condition; executes a block of statements; checks the condition; if the condition is true, repeats the block of statements until the condition becomes false

A do..while loop is very similar to a while loop. Like a while loop, a do..while loop continues repeating as long as certain conditions are true. Unlike while loops, however, those conditions are tested at the bottom of each loop. If the condition is true, the loop repeats. When the condition is not true, the loop stops.

The general format for a do..while loop is as follows:

```
do
{
block of statements
} while ( condition);
```

The following statements set up a loop that looks for an apple. This script does the same thing as the script in the preceding section that uses a while loop:

```
$fruit = array ( "orange", "apple", "grape" );
```

```
$testvar = "no";
```

```
$k = 0;
```

```
do
```

```
{
if ($fruit[$k] == "apple" )
```

```
{
$testvar = "yes";
```

```
echo "apple\n";
```

```
}
```

```
else
```

```
{
echo "$fruit[$k] is not an apple\n";
```

```
}
```

```
$k++;
```

```
} while ( $testvar != "yes" );
```

The output of these statements in a browser is as follows:

```
orange is not an apple
```

```
apple
```

This is the same output shown for the while loop example. The difference between a while loop and a do..While loop is where the condition is checked. In a while loop, the condition is checked at the top of the loop. Therefore, the loop will never execute if the condition is never true. In the do..while loop, the condition is checked at the bottom of the loop. Therefore, the loop always executes at least once, even if the condition is never true.

For example, in the preceding loop that checks for an apple, suppose the original condition is set to yes, instead of no, by using this statement:

```
$testvar = "yes";
```

The condition tests false from the beginning. It is never true. In a while loop, there is no output. The statement block never runs. However, in a do..while loop, the statement block runs once before the condition is tested. Thus, the while loop produces no output, but the do..while loop produces the following output:

```
orange is not an apple
```

The do..while loop produces one line of output before the condition is tested. It does not produce the second line of output because the condition tests false.

1.12 Comments in PHP

A comment is something which is ignored and not read or executed by PHP engine or the language as part of a program and is written to make the code more readable and understandable. These are used to help other users and developers to describe the code and what it is trying to do. It can also be used in documenting a set of code or part of a program.

PHP supports two types of comment:

1.12.1 Single Line Comment: As the name suggests these are single line or short relevant explanations that one can add in their code. To add this, we need to begin the line with (//) or (#).

Example:

```
<?php
// This is a single line comment
// These cannot be extended to more lines
echo "hello world!!!";

# This is also a single line comment
?>
```

Output:
hello world!!!

1.12.2 Multi-line or Multiple line Comment: These are used to accommodate multiple lines with a single tag and can be extended to many lines as required by the user. To add this, we need to begin and end the line with (/*...*/)

```
<?php
/* This is a multi line comment
   In PHP variables are written by adding a $ sign at the
   beginning.*/

$geek = "hello world!";
echo $geek;
?>
```

Output:
hello world!

1.13 Pre-defined Functions in PHP (Built in Functions):

Built in functions are functions that exist in PHP installation package. These built in functions are what make PHP a very efficient and productive scripting language. The built in functions can be classified into many categories. Below is the list of the categories.

1.13.1 String Functions(Manipulating Strings):

PHP provides many built-in functions for manipulating strings. Using PHP functions, you can find substrings or characters, replace part of a string with different characters, take a string apart, count the length of a string, and perform many other string manipulations.

Often you want to remove blank spaces before or after a string. You can remove leading or trailing spaces by using the following statements:

```
$string = trim($string) # removes leading & trailing spaces
```

```
$string = ltrim($string) # removes leading spaces
```

```
$string = rtrim($string) # removes trailing spaces
```

PHP can help you split a string into words, which is often handy. The general form of this function is as follows:

```
str_word_count("string",format)
```

In this expression, *format* can be 1, meaning return the words as a numeric array; or 2, meaning return the words as an array where the key is the position of the first character of the word. If you don't include a format, the function returns the number of words.

The table below shows the common PHP string functions

Function	Description	Example	Output
Strtolower	Used to convert all string characters to lower case letters	echo strtolower('Benjamin');	outputs benjamin
Strtoupper	Used to convert all string characters to upper case letters	echo strtoupper('george w bush');	outputs GEORGE W BUSH
Strlen	The string length function is used to count the number of character in a string. Spaces in between characters are also counted	echo strlen('united states of america');	24
Explode	Used to convert strings into an array variable	<pre>\$settings = explode(';', "host=localhost; db=sales; uid=root; pwd=demo"); print_r(\$settings);</pre>	Array ([0] => host=localhost [1] => db=sales [2] => uid=root [3] => pwd=demo)
Substr	Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned.	<pre>\$my_var = 'This is a really long sentence that I wish to cut short'; echo substr(\$my_var,0, 12).'...';</pre>	This is a re...
str_replace	Used to locate and replace specified string values in a given string. The function accepts three arguments. The first argument is the text to be replaced, the second argument is	echo str_replace ('the', 'that', 'the laptop is very expensive');	that laptop is very expensive

Function	Description	Example	Output
	the replacement text and the third argument is the text that is analyzed.		
Strpos	Used to locate the and return the position of a character(s) within a string. This function accepts two arguments	echo strpos('PHP Programing','Pro');	4
sha1	Used to calculate the SHA-1 hash of a string value	echo sha1('password');	5baa61e4c 9b93f3f0 682250b6cf8331b 7ee68fd8
md5	Used to calculate the md5 hash of a string value	echo md5('password');	9f961034ee 4de758 baf4de09ceeb1a75
str_word_count	Used to count the number of words in a string.	echo str_word_count ('This is a really long sentence that I wish to cut short');	12
Ucfirst	Make the first character of a string value upper case	echo ucfirst('respect');	Outputs Respect
Lcfirst	Make the first character of a string value lower case	echo lcfirst('RESPECT');	Outputs rESPECT

Numeric Functions

Numeric functions are function that return numeric results.Numeric php function can be used to format numbers, return constants, perform mathematical computations etc.**The table below shows the common PHP numeric functions**

Function	Description	Example	Output
is_number	Accepts an argument and returns true if its numeric and false if it's not	<pre><?php if(is_numeric("guru")) { echo "true"; } else { echo "false"; } ?></pre>	false
		<pre><?php if(is_numeric (123)) { echo "true"; } else { echo "false"; }</pre>	true

Function	Description	Example	Output
		} ?>	
number_format	Used to format a numeric value using digit separators and decimal points	number_format(2509663);	2,509,663
Rand	Used to generate a random number.	echo rand();	Random number
Round	Round off a number with decimal points to the nearest whole number.	echo round(3.49);	3
Sqrt	Returns the square root of a number	echo sqrt(100);	10
Cos	Returns the cosine	echo cos(45);	0.52532198881773
Sin	Returns the sine	echo sin(45);	0.85090352453412
Tan	Returns the tangent	echo tan(45);	1.6197751905439
Pi	Constant that returns the value of PI	echo pi();	3.1415926535898

1.13.2 Dates and Times

Dates and times can be important elements in a script. PHP has the ability to recognize dates and times and handle them differently than plain character strings. The computer stores dates and times in a format called a *timestamp*, which is expressed entirely in seconds. However, because this is an impractical format for humans to read, PHP converts dates from your notation into a timestamp the computer understands and from a timestamp into a format that is familiar to people. PHP handles dates and times by using built-in functions.

The timestamp format is a UNIX Timestamp, an integer that is the number of seconds from January 1, 1970 00:00:00 GMT to the time represented by the timestamp. This format makes it easy to calculate the time between two dates — just subtract one timestamp from the other.

Formatting dates

The function you will use most often is `date`. The `date` function converts a date or time from the timestamp format into a format you specify. The general format is as follows:

```
$mydate = date("format", $timestamp);
```

`$timestamp` is a variable with a timestamp stored in it. You previously stored the timestamp in the variable by using a `time` or `mktime`, as described in the next section. If `$timestamp` is not included, PHP obtains the current time from the operating system. Thus, you can get today's date with the following statement:

```
$today = date("Y/m/d");
```

If today is March 10, 2004, this statement returns:

2004/03/10

The *format* is a string that specifies the date format you want stored in the variable. For example, the format “y-m-d” returns 04-3-10, and “M.d.Y” returns Mar.10.2004. Table 5-4 lists some of the symbols that you can use in the format string. The parts of the date can be separated by hyphens (-), dots(.), slashes (/), or spaces.

Symbol	Meaning	Example
M	Month in text, abbreviated	Jan
F	Month in text not abbreviated	January
M	Month in numbers with leading zeros	02 or 12
N	Month in numbers without leading zeros	1 or 12
D	Day of the month; two digits with leading zeros	01 or 14
J	Day of the month without leading zeros	3 or 30
L	Day of the week in text not abbreviated	Friday
D	Day of the week in text as an abbreviation	Fri
W	Day of the week in numbers from 0 (Sunday) to 6 (Saturday)	5
Y	Year in four digits	2004
Y	Year in two digits	04
G	Hour between 0 and 12 without leading zeros	2 or 10
G	Hour between 0 and 24 without leading zeros	2 or 15
H	Hour between 0 and 12 with leading zeros	01 or 10
H	Hour between 0 and 12 with leading zeros	00 or 23
I	Minutes	00 or 59
S	Seconds	00 or 59
A	am or pm in lowercase	am
A	AM or PM in uppercase	AM
U	Unix seconds	1056244941

Storing a timestamp in a variable

You can assign a timestamp with the current date and time to a variable with the following statement:

```
$today = time();
```

Another way to store a current timestamp is with the following statement:

```
$today = strtotime(“today”);
```

You can store a specific date and time as a timestamp by using the function `mktime`. The format is

```
$importantDate = mktime(h,m,s,mo,d,y);
```

where *h* is hours, *m* is minutes, *s* is seconds, *mo* is month, *d* is day, and *y* is year. For example, you would store the date January 15, 2003, by using the following statement:

```
$importantDate = mktime(0,0,0,1,15,2003);
```

You can also store specific timestamps by using `strtotime` with various keywords and abbreviations that are very much like English. For instance, you can create a timestamp for January 15, 2003, as follows:

```
$importantDate = strtotime(“January 15 2003”);
```

`strtotime` recognizes the following words and abbreviations:

- _ **Month names:** Twelve month names and abbreviations
- _ **Days of the week:** Seven days and some abbreviations
- _ **Time units:** Year, month, fortnight, week, day, hour, minute, second; am, pm
- _ **Some useful English words:** Ago, now, last, next; this, tomorrow, yesterday

_ **Plus and minus:** + or -

_ **All numbers**

_ **Time zones:** For example, gmt (Greenwich Mean Time), pdt (Pacific Daylight Time), and akst (Alaska Standard Time)

You can combine the words and abbreviations in a variety of ways. The following statements are all valid:

```
$importantDate = strtotime("tomorrow"); #24 hours from now
```

```
$importantDate = strtotime("now + 24 hours");
```

```
$importantDate = strtotime("last saturday");
```

```
$importantDate = strtotime("8pm + 3 days");
```

```
$importantDate = strtotime("2 weeks ago"); # at current time
```

```
$importantDate = strtotime("next year gmt"); #1 year from now
```

```
$importantDate = strtotime("tomorrow 4am");
```

You can find differences between timestamps by using subtraction. For example, if \$importantDate is in the past and you want to know how long ago \$importantDate was, you can subtract it from the variable \$today you defined earlier. For example:

```
$timeSpan = $today - $importantDate;
```

This gives you the number of seconds between the important date and today.

You can also use the following statement to find out how many hours have transpired since the important date:

```
$timeSpan=(( $today - $importantDate)/60)/60;
```

Questions

- | | | |
|----|---|---|
| 1 | Define PHP. What these tags specify in PHP <?php and ?> | 5 |
| 2 | Can we run PHP from HTML file? If yes, how? | 5 |
| 3 | Why PHP is known as scripting language? | 5 |
| 4 | Write a program in PHP to calculate Square Root of a number. | 5 |
| 5 | List different data types and explain with example. | 5 |
| 6 | Explain comments in PHP | 5 |
| 7 | List different operators and explain with example. | 5 |
| 8 | Write the name of PHP functions that can be used to build a function that accepts any number of arguments | 5 |
| 9 | Explain in details predefined functions | 5 |
| 10 | Explain various date and time formats. | 5 |
| 11 | Write a PHP script to compute factorial of n using while or for loop | 5 |
| 12 | Write a PHP script to display Fibonacci of length 10 | 5 |
| 13 | Write a short note on Scope of Variables | 5 |
| 14 | List different loops used in PHP in details | 5 |
| 15 | List different decision making used in PHP in details | 5 |
| 16 | What is the function of for-each construct in PHP? | 5 |
| 17 | Explain various Math function available in PHP. | 5 |
| 18 | Differentiate While and Do-While statement | 5 |

Chapter 2. PHP FORM HANDLING

Learning Objectives:

After successful completion of this unit, you will be able to

6. Summarize strings data type in PHP
7. Summarize arrays in PHP.
8. Understand GET ,POST and REQUEST methods.
9. Learn and evaluate fields reading from HTML.
10. Paraphrase PHP validation.

This chapter will introduce you to strings and arrays in PHP. This chapter will let you poke around the different methods like GET, POST and REQUEST and also help retrieve data from HTML form. Learn validations involved in PHP.

2.1 Strings

In any programming or scripting language, a string is a sequence of characters written in Single quote or Double quotes. For example, "Hello Friend!" is a string written in Double quotes. In this chapter, we will see some commonly used string manipulation functions. The PHP string functions are part of the PHP core. Installation is not required to use these functions in PHP.

1) The echo() Function:

The echo() function is one of the important string functions in the PHP which is used to display one or more strings on the output window.

Syntax:

```
echo(string $str)
```

Example:

```
<?php
$str="Hello Friends!";
echo $str;
?>
```

Output:

Hello Friends!

2) The strtolower() Function::

The strtolower() function returns a string in lowercase letter.

Syntax:

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

Example:

```
<?php
$str="My Name is PARAG";
$str=strtolower($str);
echo $str;
?>
```

Output:

my name is parag

3) The strtoupper() Function:

The strtoupper() function returns a string in uppercase letter.

Syntax:

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

Example:

```
<?php
$str="My Name is Parag";
$str=strtoupper($str);
echo $str;
?>
```

Output:

MY NAME IS PARAG

4) The ucfirst() Function:

The ucfirst() function returns a string converting the first character into uppercase.

The case of other characters will not be changed.

Syntax:

```
string ucfirst (string $str)
```

Example:

```
<?php
$str="my name is Parag";
$str=ucfirst($str);
echo $str;
?>
```

Output:

My name is Parag

5) The lcfirst() Function:

The lcfirst() function returns a string converting the first character into lowercase. The case of other characters will not be changed.

Syntax:

```
string lcfirst (string $str)
```

Example:

```
<?php
$str="MY name IS Parag";
$str=lcfirst($str);
echo $str;
?>
```

Output:

mY name IS Parag

6) The ucwords() Function:

The ucwords() function returns a string converting the first character of each word into uppercase.

Syntax:

```
string ucwords (string $str)
```

Example:

```
<?php
$str="my name is parag achaliya";
$str=ucwords($str);
echo $str;
?>
```

Output:

My Name Is Parag Achaliya

7) The strrev() Function:

The strrev() function returns a reversed string.

Syntax:

```
string strev (string $string)
```

Example:

```
<?php
$str="my name is parag";
$str=strev($str);
echo $str;
?>
```

Output:

garap si eman ym

8) The strlen() Function:

The strlen() function returns the length of the string.

Syntax:

```
int strlen (string $string)
```

Example:

```
<?php
$str="my name is Parag Achaliya";
$str=strlen($str);
echo $str;
?>
```

Output:

25

9) The strcmp() Function:

String comparison is one of the most common tasks in any programming. strcmp() is a built-in string comparison function in PHP. It is case sensitive means it treats uppercase and lowercase separately. This function compares two strings and tells whether a string is greater, smaller, or equal to another string.

Syntax:

```
strcmp($str1,$str2)
```

Parameters

The `strcmp()` function requires two strings parameter. Both the parameters are mandatory to pass in the function.

Value return by `strcmp()`

This function returns an integer value based on the comparison.

It **returns 0** if both strings are **equal**, i.e., `$str1 = $str2`

It **returns negative** value if string1 is **less than** string2, i.e., `$str1 < $str2`

It **returns positive** value if string1 is **greater than** string 2, i.e., `$str1 > $str2`

Example:

```
<?php
echo strcmp("Hello ", "HELLO"). " because the first string is greater than the second
string.";
echo "</br>";
echo strcmp("Hello world", "Hello world Hello"). " because the first string is less than
the second string.";
?>
```

Output:

1 because the first string is greater than the second string.

-6 because the first string is less than the second string.

Remark: Second output has returned -6 because the first string is smaller than 6 characters by the second string, including whitespace.

String1	String2	Output	Explanation
Hello	Hello	0	Both the strings are equal.

Hello	hello	-1	String1 < String2 because the ASCII value of H is 72 and the ASCII value of h is 104 so H < h.
hello	Hello	1	String1 > String2 because the ASCII value of H is 72 and the ASCII value of h is 104 so H > h.
Hello PHP	Hello	4	String1 > String2 because the String1 is greater than the String2 by 6 characters including the whitespace.
hello	Hello PHP	1	String1 > String2 because the ASCII value of H is 72 and the ASCII value of h is 104 so H < h.
Hello	Hello PHP	-4	String1 < String2 because the String1 is smaller than the String2 by 4 characters including whitespace.

10) The print() Function:

print() function is used to print one or more strings on the output window.

Syntax:

```
int print ( string $arg );
```

Example:

```
<?php
$str = "Hello PHP!";
print $str;
print "<br>PHP is the Server Side Scripting Language";
?>
```

Output:

```
Hello PHP!
PHP is the Server Side Scripting Language
```

11) The printf() Function:

The printf() function is a predefined PHP function. It is used to print the formatted string on the output window. This function takes different parameters as format, arg1, arg2, arg++ percent (%) signs in the main format parameter.

Syntax:

```
printf(format,arg1,arg2,arg++);
```

Parameter	Description	Required/ Optional
format	<p>Specifies the string. Different possible format values are:</p> <ul style="list-style-type: none"> • %% - Returns a percent sign • %b : Binary number • %c : The character according to the ASCII value • %d : Signed decimal number (negative, zero or positive) • %e : Scientific notation using a lowercase (e.g. 1.2e+2) • %E : Scientific notation using a uppercase (e.g. 1.2E+2) • %u : Unsigned decimal number (equal to or greater than zero) • %f : Floating-point number (local settings aware) • %F : Floating-point number (not local settings aware) • %g : shorter of %e and %f • %G : shorter of %E and %f • %o : Octal number • %s : String • %x : Hexadecimal number (lowercase letters) • %X : Hexadecimal number (uppercase letters) 	Required
arg1	Argument to be inserted at first %-sign.	Required

arg2	Argument to be inserted at second %-sign.	Optional
arg++	Argument to be inserted at third, fourth, etc. %s sign	Optional

Example:

```
<?php
$version = 7;
$str = "YCMOU";
printf("We are learning PHP %u at %s.", $version, $str);
?>
```

Output:

We are learning PHP 7 at YCMOU.

12) The strpos() Function:

The strpos() is a predefined PHP function. It is used to find the position of the first occurrence of a string inside another string.

Syntax:

```
strpos(string,find,start);
```

Parameter	Description	Required/ Optional
string	Specify the string to search.	Required
find	Specify the string to find.	Required
start	Specify where to begin the search.	Optional

Example:

```
<?php
echo strpos("PHP is Server Side Scripting Language, I love PHP","PHP");
?>
```

Output:

0

13) The substr() Function:

The substr() is a built-in PHP function. This function returns a part of a string specified by the start and length parameter. This function is supported in **PHP 4** and **above** versions.

Syntax:

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

Parameter	Description	Required/ Optional
string	Specify the string to find its substring.	Required
start	Specifies from where to start extraction the string	Required
length	Length of string to be cut from the main string	Optional

Return Values

The substr() function returns an extracted part of the string on successful execution. Otherwise, it will return the FALSE or empty string on failure.

Example: 1

```
<?php
    echo substr("Hello PHPTutorial", 3). "</br>";
    echo substr("Hello PHPTutorial", 0). "</br>";
    echo substr("Hello PHPTutorial", 9). "</br>";
    echo substr("Hello PHPTutorial", -4). "</br>";
    echo substr("Hello PHPTutorial", -10). "</br>";
    echo substr("Hello PHPTutorial", -16). "</br>";
?>
```

Output:

```
lo PHPTutorial
Hello PHPTutorial
Tutorial
rial
```

HPTutorial
ello PHPTutorial

Example: 2

```
<?php
    echo substr("Hello PHPTutorial", 3, 8). "</br>";
    echo substr("Hello PHPTutorial", 0, 9). "</br>";
    echo substr("Hello PHPTutorial", 6, -4). "</br>";
    echo substr("Hello PHPTutorial", 4, -7). "</br>";
    echo substr("Hello PHPTutorial", -6, -10). "</br>";
    echo substr("Hello PHPTutorial", -6, -1). "</br>";
?>
```

Output:

lo PHPTu
Hello PHP
PHPTuto
o PHPT

toria

14) The str_word_count() Function:

The str_word_count() PHP function is used to count the number of words in a string.

Syntax:

```
str_word_count($str)
```

Example:

```
<?php
$str = "Hello My Dear Friends!";
$wcount = str_word_count($str);
echo "No. of words in $str are ".$wcount;
?>
```

Output:

No. of words in Hello My Dear Friends! are 4

2.2 Arrays

An array is a special variable in any programming or scripting language which can store more than one value at a time. For example, if you have a list of fruits, storing the fruits in single variable could look like this:

```
$fruit1 = "Banana";  
$fruit2 = "Mango";  
$fruit3 = "Apple";
```

This is possible for a small amount of list items. But what if you want to store more than 10 fruits or if you want to access a specific list item? In that case, it is not advisable to have the variables as shown in above example. The best possible solution for the said problems would be using an Array. An array can store multiple values using a single name, and you can access these values by referring to the index number.

Create an Array in PHP:

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

```
array();
```

In PHP, there are three types of arrays:

1. Indexed arrays - Arrays with a numeric index
2. Associative arrays - Arrays with named keys
3. Multidimensional arrays - Arrays containing one or more arrays

Example:

An array of fruits can be created for above example as shown below,

```
$fruits = array("Banana", "Mango", "Apple");
```

The count() Function:

The count() function of an array is used to return the length i.e. the number of elements stored in an array.

Syntax:

```
count(array);
```

Example:

```
<?php  
$fruits = array("Banana", "Mango", "Apple");
```

```
echo count($fruits);
```

```
?>
```

Output:

3

2.2.1 Indexed Arrays:

An index in an array is the numerical value assigned to every element of an array.

The Indexed Arrays in PHP can be created in two different ways,

1. The index can be assigned automatically as shown in below example. In this case, the index will automatically start from 0 by default.

```
$fruits = array("Banana", "Mango", "Apple");
```

2. The index can be assigned manually as shown in below example:

```
$fruits[0] = "Banana";
```

```
$fruits[1] = "Mango";
```

```
$fruits[2] = "Apple";
```

Example:

```
<?php
```

```
$fruits = array("Banana", "Mango", "Apple");
```

```
echo "My favourite fruits are " . $fruits[0] . ", " . $fruits[1] . " and " . $fruits[2] . " .";
```

```
?>
```

Output:

My favourite fruits are Banana, Mango and Apple.

2.2.2 Associative Arrays:

In PHP, the associative arrays are the other types of arrays that use named keys that are assigned to them. The associative arrays can be created in two different ways,

1. As shown in below example,

```
$sage = array("Parag"=>"34", "Vivek"=>"35", "Monali"=>"35");
```

2. As shown in below example,

```
$sage['Parag'] = "34";
```

```
$sage['Vivek'] = "35";
```

```
$sage['Monali'] = "35";
```

Example:

```
<?php
```

```
$age = array("Parag"=>"34", "Vivek"=>"35", "Monali"=>"35");  
echo "Parag is " . $age["Parag"] . " years old."  
?>
```

Output:

Parag is 34 years old.

2.2.3 Multidimensional Arrays:

In the previous sections, we have seen the arrays that are a single list of key and value pairs. But, sometimes you may want to store values that can have more than one key and its value. For this, we can use the multidimensional arrays. A multidimensional array is an array having one or more arrays. PHP supports multidimensional arrays that are two, three, four, or more levels deep. But, the arrays more than three levels deep are difficult to manage.

Two-dimensional Arrays:

A two-dimensional array is an array of arrays (Similarly, a three-dimensional array is an array of arrays of arrays and so on).

Example:

Consider the following table as an example,

Name	Stock	Sold
Hyundai	23	20
Honda	14	11
Maruti	7	4
Kia	11	7

We can store the above table data in a two-dimensional array like this:

```
$cars = array (  
    array("Hyundai", 23, 20),  
    array("Honda", 14, 11),  
    array("Maruti", 7, 4),  
    array("Kia", 11, 7)
```


);

Now the above two-dimensional \$cars array contains four sub arrays and it has two indices: row and column. To access the elements of the \$cars array, we must point to the two indices (row and column) as shown in below example:

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2]."<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2]."<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2]."<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2]."<br>";
?>
```

Output:

Hyundai: In stock: 23, sold: 20.
Honda: In stock: 14, sold: 11.
Maruti: In stock: 7, sold: 4.
Kia: In stock: 11, sold: 7.

Sorting Array:

The elements of an array can be sorted in alphabetical or numerical order as well as in descending or ascending order. In this section, we will see the various built-in array sorting functions available in the PHP.

1) The sort() Function:

This function will sort the array in ascending order.

Example:

```
<?php
$fruits = array("Mango", "Banana", "Apple");
sort($fruits);
$length = count($fruits);
for($i = 0; $i < $length; $i++) {
    echo $fruits[$i];
    echo ", ";
}
?>
```

Output:

Apple, Banana, Mango,

2) The rsort() Function:

This function will sort the array in descending order

Example:

```
<?php
$fruits = array("Mango", "Banana", "Apple");
rsort($fruits);
$length = count($fruits);
for($i = 0; $i < $length; $i++) {
    echo $cars[$i];
    echo ", ";
}
?>
```

Output:

Mango, Banana, Apple,

3) The asort() Function:

This function will sort the associative arrays in ascending order, according to the value.

Example:

```
<?php
$age = array("Parag"=>"34", "Vivek"=>"36", "Monali"=>"35");
asort($age);
foreach($age as $i => $i_value) {
    echo "Key=" . $i . ", Value=" . $i_value;
    echo "<br>";
}
?>
```

Output:

Key=Parag, Value=34
Key=Monali, Value=35
Key=Vivek, Value=36

4) The ksort() Function:

This function will sort the associative arrays in ascending order, according to the key.

Example:

```
<?php
$age = array("Parag"=>"34", "Vivek"=>"36", "Monali"=>"35");
ksort($age);
foreach($age as $i => $i_value) {
    echo "Key=" . $i . ", Value=" . $i_value;
    echo "<br>";
}
?>
```

Output:

```
Key=Monali, Value=35
Key=Parag, Value=34
Key=Vivek, Value=36
```

5) The arsort() Function:

This function will sort the associative arrays in descending order, according to the value.

Example:

```
<?php
$age = array("Parag"=>"34", "Vivek"=>"36", "Monali"=>"35");
arsort($age);
foreach($age as $i => $i_value) {
    echo "Key=" . $i . ", Value=" . $i_value;
    echo "<br>";
}
?>
```

Output:

```
Key=Vivek, Value=36
Key=Monali, Value=35
Key=Parag, Value=34
```

6) The krsort() Function:

This function will sort the associative arrays in descending order, according to the key.

Example:

```
<?php
$age = array("Parag"=>"34", "Vivek"=>"36", "Monali"=>"35");
krsort($age);
foreach($age as $i => $i_value) {
    echo "Key=" . $i . ", Value=" . $i_value;
    echo "<br>";
}
?>
```

Output:

```
Key=Vivek, Value=36
Key=Monali, Value=35
Key=Parag, Value=34
```

2.3 GET, POST and REQUEST Methods:

The Hypertext Transfer Protocol (HTTP) is designed to establish the communications between clients and servers. It works on request-response protocol between client and server. For example: If a client, let's say a browser, sends an HTTP request to the server; then that server returns a response to its client. The response will contain the status information about the request and may also contain the requested content of the client. Various HTTP methods are responsible for establishing the communication between such clients and the server.

HTTP Methods:

- GET
- PUT
- POST
- HEAD
- PATCH
- DELETE
- OPTIONS

The two most common HTTP methods are: GET and POST.

2.3.1 GET Method:

The GET method is used to request the data from the specified resource. It is one of the most commonly used HTTP methods in HTML forms. In case of GET method, the query string of name/value pairs is sent in the URL like this:

/sample/test_form.php?name1=value1&name2=value2

The GET method sends the user information in encoded form appended with the page request. This page and the encoded information are separated by the ? character as shown in above example.

Some important points about GET method:

- The GET method produces a long string that appears in server logs.
- The GET method can send upto 1024 characters only.
- If you want to send a password or other sensitive information to the server, in that case never use the GET method.
- Binary data, like images or word documents, can't be sent using the GET method.
- PHP provides a \$_GET array to access all the sent information using the GET method.

Example:

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

}?>

Output:



Fig. 2.1: HTML Form using GET Method [1]

2.3.2 POST Method:

The POST method is used to send the data to the server. It is used to create or update the resource. The data sent to the server using POST method is stored in the request body of the HTTP request. Like the GET method, POST method is also one of the most common HTTP methods.

Some important points about POST method:

- The POST method can send the data of any size.
- The POST method can send ASCII as well as binary data.
- HTTP header is used to send the data by POST method. So the data security depends on the HTTP protocol. By using Secure HTTP, the information security can be obtained.
- PHP provides a `$_POST` array to access all the sent information using POST method.

Example:

```
<html>
<body>
  <form action = "<?php $_PHP_SELF ?>" method = "POST">
    Name: <input type = "text" name = "name" />
    Age: <input type = "text" name = "age" />
    <input type = "submit" />
  </form>
</body>
</html>
<?php
if( $_POST["name"] || $_POST["age"] ) {
  if (preg_match("/^[A-Za-z'-]"/,$_POST['name'] )) {
    die ("Invalid name, it should be alpha");
  }
  echo "Welcome ". $_POST['name']. "<br />";
```

```

        echo "You are ". $_POST['age']. " years old.";
        exit();
    }
?>

```

Output:



The screenshot shows a web form with a light gray border. Inside, the text 'Name:' is followed by a text input field. To the right, 'Age:' is followed by another text input field. Further right is a button labeled 'Submit'.

Fig. 2.2: HTML Form using POST Method [1]

Difference between GET and POST Methods:

Parameter	GET	POST
BACK button/ Reload	Harmless	Data will be re-submitted
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Can not be cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters are saved in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes. (Maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure because data	POST is a little safer because the

	sent is part of the URL. (Never use GET when sending passwords or other sensitive information)	parameters are not stored in browser history or in web server logs.
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

2.3.3. REQUEST Method:

The `$_REQUEST` variable of PHP contains the contents of `$_GET`, `$_POST`, and `$_COOKIE`. It can be used to get the result from form data sent using the GET and POST methods of PHP.

Example:

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

Output:



The screenshot shows a web form with two text input fields. The first field is labeled "Name:" and the second is labeled "Age:". To the right of the "Age:" field is a button labeled "Submit". The form is enclosed in a light gray border.

Fig. 2.3: HTML Form using REQUEST Method [1]

2.4 Reading Fields from HTML

The `$_POST` is a PHP super global variable. It is used to collect the form data after submitting an HTML form. This can be done with the `method="post"` and the `$_POST` is used to pass the variables.

Example:

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

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

In the above example, a form with an input field and a submit button is created using HTML tags. When any user will submit the data by clicking on the "Submit" button, the form data is sent to the specified file mentioned in the action attribute of the `<form>`. In this example, the file has been pointed to itself for processing the form data. If you want to use another PHP file to process form data, replace that file with the filename of your choice. After that, the super global variable `$_POST` can be used to collect the value of the input field.

The \$_GET is also a PHP super global variable like \$_POST. It is also used to collect the form data after submitting an HTML form. This can be done with the method="get" and the \$_GET is used to pass the variables. The above example can also be used to the fields from HTML just by replacing post by get everywhere.

2.5 PHP Validations:

An HTML form can contain various input fields such as text box, radio buttons, checklist, checkbox, submit button, etc. These input fields should be validated which ensures that the user has entered some information in all the required fields.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

1. Empty String
2. Validate String
3. Validate Numbers
4. Validate Email
5. Validate URL

2.5.1 Empty String:

An error message will be shown on output if the user will leave the required field empty. The following example checks whether the field is empty or not.

```
if (empty ($_POST["name"]))
{
    $errMsg = "Error! Name can not be empty.";
    echo $errMsg;
}
else
{
    $name = $_POST["name"];
}
```

2.5.2 Validate String:

The following code will check that the field will contain only alphabets and whitespace, for example - name. An error message will be shown if the name field does not receive valid input from the user.

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

2.5.3 Validate Number:

The following code will validate that the field should only contain a numeric value. For example, Mobile no. If the Mobile no field does not receive numeric data from the user, the code will display the error message.

```
$mobilenos = $_POST ["mobile_no"];  
if (!preg_match ("/^[0-9]*$/", $mobilenos) )  
{  
    $ErrMsg = "Only numeric value is allowed in mobile no.";   
    echo $ErrMsg;  
}  
else  
{  
    echo $mobilenos;  
}
```

2.5.4 Validate Email:

A valid email must have @ and . symbols. PHP provides various methods to validate the email address. But here, we will use regular expressions to validate the email address. The following code will validate the email address provided by the user through HTML form and an error message will be displayed if the field does not contain a valid email address.

```
$email = $_POST ["email"];
$pattern = "^[_a-z0-9-]+(\\.[_a-z0-9-]+)*@[a-z0-9-]+(\\.[a-z0-9-]+)*(\\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) )
{
    $ErrMsg = "Invalid Email Id.";
    echo $ErrMsg;
}
else
{
    echo "Your valid email address is: " . $email;
}
```

2.5.5 Validate URL:

The following code will validate the URL of the website provided by the user in HTML form. If the field does not contain a valid URL, the code will display an error message, i.e., "Invalid URL".

```
$websiteURL = $_POST["website"];
if (!preg_match("/^b(?:(:https?|ftp):\\/\\/[www\\.])[-a-z0-9+&@#\\/%?~_!|:,;]*[-a-z0-9+&@#\\/%?~_]/i",$website))
{
    $websiteErr = "Invalid URL";
    echo $websiteErr;
}
else
{
    echo "Website URL is: " . $websiteURL;
}
```

Questions:

- | | | |
|----|--|---|
| 1 | How can we create links in PHP pages? | 5 |
| 2 | Difference between include and require? | 5 |
| 3 | Difference between GET and POST methods. | 5 |
| 4 | What is the purpose of “nl2br “string function? | 5 |
| 5 | Write a program to do string manipulation | 5 |
| 6 | Write a short note on REQUEST method | 5 |
| 7 | Write a short note on GET method | 5 |
| 8 | Write a short note on POST method | 5 |
| 9 | Write a short note on Use of substring function. | 5 |
| 10 | What is the purpose of \$_PHP_SELF? | 5 |
| 11 | Create HTML form to enter one number. Write PHP code to display the message about number is odd or even. | 5 |
| 12 | Write a PHP program to accept a positive integer ‘N’ through a HTML form and to display the sum of all the numbers from 1 to N | 5 |
| 13 | Write a short note on array. | 5 |
| 14 | Short note on PHP validation | 5 |
| 15 | Write a PHP code for Book store to apply all validation. | 5 |

Chapter 3. File Handling, Session, Cookies in PHP

Learning Objectives:

After successful completion of this unit, you will be able to

11. Summarize file handling in PHP

12. Understand session in PHP.

13. Learn cookies in PHP.

This chapter will introduce you to file handling in PHP. You will learn how to create, open, read, write, upload and delete files in PHP.

This chapter will let you poke around the different sessions, cookies and filters used in PHP.

3.1 Introduction:

Any web application needs to be able to handle files. For various tasks, you will often need to open and process a file. We can use the PHP File System to create files, read them line by line, character by character, write them, append them, remove them, and close them. This can be done using various PHP Functions.

3.2 File Open/Read

In this section, we will see various file open or file read functions available in PHP. All these functions are built-in in PHP.

3.2.1 fopen():

The PHP fopen() function opens a file or a URL and returns the resource. The file name and mode parameters are passed to the fopen() function. File name denotes the file to be opened, while mode denotes the file mode, such as read-only, read-write, or write-only.

Syntax:

```
$file = fopen("filename", "mode")
```

The name of the file to be opened is the first parameter of fopen(), and the second parameter defines the mode in which the file should be opened. You can open the file in one of the following ways:

Mode	Meaning	Description
r	Open a file for read only.	File pointer begins at the start of the file
w	Open a file for write only.	Removes the file's contents or produces a new one if it doesn't exist. The file pointer begins at the start of the file.

a	Open a file for write only.	The data in the file is maintained. The file pointer begins at the file's end. If the file does not exist, it is created.
x	Create a new file for write only.	If the file already exists, returns FALSE and an error.
r+	Open a file for read/write.	The file pointer begins at the start of the file.
w+	Open a file for read/write.	Removes the file's contents or creates a new one if it doesn't exist. The file pointer begins at the start of the file.
a+	Open a file for read/write.	The data in the file is maintained. The file pointer begins at the file's end. If the file does not exist, it is created.
x+	Creates a new file for read/write.	If the file already exists, returns FALSE and an error.

3.2.2 fread():

The fread() function reads data from a file that is currently open. The first parameter of fread() is the filename to read from, and the second is the maximum number of bytes to read. Following PHP code finishes reading the file:

Syntax:

```
fread(file,file_size);
```

3.2.3 fclose():

To close an open file, use the fclose() function. Closing all files once you've done with them is good programming practise. You don't want an open file hogging system resources on your server! The filename or a variable containing the filename we want to close is required by fclose().

Syntax:

```
fclose(filename);
```

3.2.4 fgets():

A single line from a file is read using the fgets() function.

Syntax:

```
fgets(filename);
```

3.2.5 feof():

The feof() function determines whether or not the file has reached the "end-of-file" (EOF). For looping through data of unknown length, the feof() function is useful.

Syntax:

```
feof(filename);
```

3.2.6 fgetc():

A single character from a file is read using the fgetc() function. The file pointer moves to the next character after the fgetc() function is called.

Syntax:

```
fgetc(filename);
```

3.3 File Create/Write:

In this section, we will see various functions available in PHP which are used to create or write the files. All these functions are built-in in PHP.

3.3.1 fopen():

A file can also be created using the fopen() function. In PHP, a file is created using the same feature that opens files, which can be a little confusing. If you call fopen() on a file that doesn't exist, it will create it if it's being used for writing (w) or appending (a).

Syntax:

```
$file = fopen("samplefile.txt", "w")
```

3.3.2 fwrite():

To write to a file, the fwrite() function is used. The name of the file to be written to is the first parameter of fwrite(), and the string to be written is the second.

Syntax:

```
fwrite(file, text);
```

Example:

```
<?php
$file = fopen("samplefile.txt", "w");
$text = "Parag Achaliya\n";
fwrite($file, $text);
$text = "Vivek Patil\n";
fwrite($file, $text);
fclose($file);
?>
```

Output:

Parag Achaliya

Vivek Patil

3.4 File Deletion:

The file on your computer can easily be deleted using the Delete button or Shift + Delete. By simply doing Delete operation, the file will be deleted from the current location & will be moved to the Recycle Bin. But by doing Shift + Delete operation, the file will be permanently deleted from the computer. If you want to delete the file using PHP then the unlink() function will be used. The unlink() function removes a file from the system.

Syntax:

unlink(file, context)

Parameter Details:

Parameter	Description
file	Required. Specifies the file to delete with its path
context	Optional. The meaning of the file handle is defined. Context is a collection of choices that can change how a stream behaves.

Example:

```
<?php
$samplefile = fopen("file.txt","w");
echo fwrite($samplefile,"Hello Friends!");
fclose($samplefile);

unlink("test.txt");
?>
```

It returns TRUE upon successful file deletion else FALSE on failure. It works with a 4.0 or higher version of PHP.

3.5 File Upload:

3.5.1 Upload New File

Uploading files to the server is simple with PHP. However, convenience can be dangerous, so be cautious while allowing file uploads! We will follow some important steps for uploading using PHP.

Step 1: First, make sure PHP is set up to accept file uploads. Look for the file uploads directive in your "php.ini" file and turn it on like this,

```
file_uploads = On
```

Step 2: Create an HTML form that allows users to upload any file they want.

```
<html>
    <body>
        <form action="upload.php" method="post">
            Upload file:
            <input type="file" name="FileUpload" id="FileUpload">
            <input type="submit" value="Upload" name="button">
        </form>
    </body>
</html>
```

Step 3: Create an "upload.php" file containing the following code of uploading a file.

```
<?php
$trgt_dir = "uploads/";
$trgt_file = $trgt_dir . basename($_FILES["FileUpload"]["name"]);
$OkUpload = 1;
$FileType = strtolower(pathinfo($trgt_file,PATHINFO_EXTENSION));
if(isset($_POST["button"]))
{
    $CheckFile = getimagesize($_FILES["FileUpload"]["temp_name"]);
    if($CheckFile !== false)
    {
        echo "Valid image - " . $check["mime"] . ".";
        $OkUpload = 1;
    }
    else
    {
        echo "Invalid image file.";
        $uploadOk = 0;
    }
}
?>
```

Explanation:

- \$trgt_dir = "uploads/" - specifies the folder where file is uploaded
- \$trgt_file specifies the location of the to-be-uploaded file

- \$OkUpload=1 for future use
- \$FileType - The file extension is stored in this variable (in lowercase)
- Next, determine if the image file is a genuine image or a forgery.

Note - In the directory where the "upload.php" file is located, create a new directory called "uploads." The files you've uploaded will be saved there.

3.5.2 Check if File already Exists:

We'll start by seeing if the to-be-upload file is already available in the "uploads" folder. If the file is available, an error message will be shown, and the value of \$OkUpload is set to 0 as shown in below code,

```
if (file_exists($trgt_file))
{
    echo "The file is already exists.";
    $OkUpload = 0;
}
```

3.5.3 Limit the File Type:

Users can only upload JPG, JPEG, PNG, and GIF files using the code below. Before setting \$OkUpload to 0 for all other file types, you'll get an error message.

```
if($FileType!="jpg"    &&    $FileType!="png"    &&    $FileType!="jpeg"
&&$FileType!="gif")
{
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $OkUpload = 0;
}
```

3.6 Cookies:

To recognise a user, a cookie is frequently used. A cookie is a tiny file placed on the user's machine by the server. The cookie will be sent each time the same machine requests a page via a browser. Cookie values can be created and retrieved using PHP.

3.6.1 Create Cookies:

The setcookie() function is used to create the cookie.

Syntax:

```
setcookie(cookie_name, cookie_value, expire, path, domain, secure, http_only);
```

In above syntax, the name parameter is the only one that needs to be filled out. The rest of the parameters are entirely up to you.

3.6.2 Create/Retrieve Cookie:

Consider the following example of creating the cookie. In this example, the “user” cookie with “Parag Achaliya” value is created. This cookie will expire after 20 days (86400 * 20). The “/” indicates that the cookie will be available on the whole website otherwise you can select the directory you want. Then we will use the global variable \$_COOKIE to retrieve the value of “user” cookie. We will also use the isset() function to check whether the cookie is set. Remember, the setcookie() function must be used before the <html> tag.

```
<?php
    $name = "user";
    $value = "Parag Achaliya";
    setcookie($name, $value, time() + (86400 * 20), "/");
?>
<html>
    <body>
        <?php
            if(!isset($_COOKIE[$name]))
            {
                echo "Cookie '" . $name . "' is not set!";
            }
            else
            {
                echo "Cookie '" . $name . "' is set!<br>";
                echo "It's value is: " . $_COOKIE[$name];
            }
        ?>
    </body>
</html>
```

3.6.3 Modify Cookie:

Again set the cookie by using the setcookie() function to modify it.

```
<?php
    $name = "user";
    $value = "Vivek Patil";
    setcookie($name, $value, time() + (86400 * 20), "/");
?>
<html>
    <body>
```

```

        <?php
            if(!isset($_COOKIE[$name]))
            {
                echo "Cookie '" . $name . "' is not set!";
            }
            else
            {
                echo "Cookie '" . $name . "' is set!<br>";
                echo "It's value is: " . $_COOKIE[$name];
            }
        ?>
    </body>
</html>

```

3.6.4 Delete Cookie:

Cookies can be deleted by again using the `setcookie()` function with expiry date as shown in following example,

```

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

```

3.7 Sessions:

A session is a method of storing data in variables that can be used across several pages. The data is not saved on the user's screen, unlike a cookie. When working with a programme, you open it, make changes, and then close it. This is similar to a Session. The computer recognises you. It knows when you start and stop using the programme. However, there is an issue on the internet: the web server has no idea who you are or what you do because the HTTP address does not keep track of state. Session variables address this issue by storing user data that can be used across several pages. Session variables are stored in the browser before the user closes it. As a result, session variables store information about a single user and are accessible from all pages in a single programme.

3.7.1 Start Session:

The `session_start()` function is used to start the session. The PHP global variable `$_SESSION` is used to set session variables. Now, let's create a new PHP page called

"session_example1.php". We'll start a new PHP session and set some session variables in this section. Remember that the program must start with the session_start() function. Even before the HTML tags.

```
<?php
    session_start();
?>
<html>
    <body>
        <?php
            $_SESSION["fav_color"] = "Brown";
            $_SESSION["fav_animal"] = "Lion";
            echo "Session variables are set.";
        ?>
    </body>
</html>
```

3.7.2 Get the Values of Session Variable:

Now we will create one more PHP page as "session_example2.php". We'll access the session data we set on the first page "session_example1.php" from this page. Session variables are retrieved from the session we open at the start of each page, rather than being moved to each new page individually. The global variable \$_SESSION stores all session variable values.

```
<?php
    session_start();
?>
<html>
    <body>
        <?php
            // Print session variables which were set in previous example
            echo $_SESSION["fav_color"] . " is favorite color.<br>";
            echo $_SESSION["fav_animal"] . " is favorite animal.";
        ?>
    </body>
</html>
```

3.7.3 Modify Session Variable:

If you want to modify the session variable then simply overwrite the values of those variables. Consider the following example. In this example, the value of favourite color is changed from Brown to Red by simply overwriting it.

```
<?php
```

```

        session_start();
    ?>
<html>
    <body>
        <?php
            // to change a session variable, just overwrite it
            $_SESSION["fav_color"] = "Red";
            echo $_SESSION["fav_color"].
        ?>
    </body>
</html>

```

3.7.4 Delete Session:

The `session_unset()` and `session_destroy()` functions are used to delete all global session variables and destroy the session. Consider the following example,

```

<?php
    session_start();
?>
<html>
    <body>
        <?php
            session_unset();    // delete all session variables
            session_destroy();  // destroy the session
        ?>
    </body>
</html>

```

3.8 Filters:

External input is validated and sanitised using PHP filters. Validating the data is the process of determining whether the data is in proper format whereas Sanitizing the data is the process of removing any illegal character from that data. Many of the functions needed for verifying user input are included in the PHP filter extension, which is meant to make data validation easier and faster.

3.8.1 Why to use the filters in PHP?

External input is used by many web applications. Those External inputs can be:

- Cookies
- Server variables
- Web services data
- Results of Database queries
- User inputs submitted via a form

These external data should always be validated. Any invalid data might cause security issues and cause your website to crash. You can ensure that your application receives the correct input by using the PHP filters.

3.8.2 Sanitize the String:

The process of validating the data and sanitizing the data is done by `filter_var()` function of PHP. The `filter_var()` function applies a specified filter to a single variable. This function takes two parameters:

- The variable you want to check
- The type of check to use

Example:

```
<?php
$string = "<h2>Hello Friends!</h2>";
$new_string = filter_var($string, FILTER_SANITIZE_STRING);
echo $new_string;

?>
```

In the above example, `filter_var()` function with the value `FILTER_SANITIZE_STRING` will remove all the HTML tags from the string and will return the plain string.

3.8.3 Validate the Integer:

Consider the following example. In this example, `filter_var()` function is used to validate the integer using the `FILTER_VALIDATE_INT` parameter. If the value of variable is an integer then the program will output: "Valid Integer" else it will output: "Invalid Integer".

```
<?php
$var = 7;
if (!filter_var($var, FILTER_VALIDATE_INT) === false)
{
    echo("Valid Integer");
}
else
{
    echo("Invalid Integer");
}

?>
```

In the above example, one problem may occur if we set the value of `$var` to 0. It will show the output as "Invalid Integer" which is not correct. To solve this problem, we need to do some additional programming as shown in below code,


```

<?php
    $var = 0;
    if (filter_var($var, FILTER_VALIDATE_INT) === 0 || !filter_var($var,
    FILTER_VALIDATE_INT) === false)
    {
        echo("Valid Integer");
    }
    else
    {
        echo("Invalid Integer");
    }
?>

```

3.8.4 Validate IP Address:

The filter_var() function can also be used to check if the IP address is valid or not. This can be done using the FILTER_VALIDATE_IP value in the filter_var() function as shown in below code,

```

<?php
    $ip_address = "192.168.1.1";
    if (!filter_var($ip_address, FILTER_VALIDATE_IP) === false)
    {
        echo("IP address is valid.");
    }
    else
    {
        echo("IP address is invalid.");
    }
?>

```

3.8.5 Sanitize and Validate an Email Address:

The filter_var() function will first sanitize the variable i.e. it will remove all illegal characters then it will check whether the email address is valid. This can be done using the following code,

```

<?php
    $email_id = "achaliya.pncoe@snjb.org";

    $email = filter_var($email_id, FILTER_SANITIZE_EMAIL);

    if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false)
    {
        echo("Email address is valid.");
    }

```

```

    }
    else
    {
        echo("Email address is invalid.");
    }
?>

```

3.8.6 Sanitize and Validate a URL:

The `filter_var()` function will first sanitize the variable i.e. it will remove all illegal characters then it will check whether the URL is valid. This can be done using the following code,

```

<?php
$url = "https://paragnachaliya.in/";

$url = filter_var($url, FILTER_SANITIZE_URL);

if (!filter_var($url, FILTER_VALIDATE_URL) === false)
{
    echo("URL is valid.");
}
else
{
    echo("URL is invalid.");
}
?>

```

Unit 4

Errors and Exception Handling in PHP

4.1 Learning Objectives

After successful completion of this unit, students will be able to handle Errors as well as Exception occurred at the time of execution of source code.

4.2 Introduction

An error is an action which is wrong or inappropriate. Sometimes, an error can be the synonymous with a mistake. In statistics, "error" refers to the difference between the computed value and the correct value. An error could result in failure. Whereas an unexpected program result is an exception and it can be handled by program itself. In this chapter we will see process of errors and exceptions handling in PHP.

4.3 Compilation of Errors and Warning

Generally Error handling process is sequential process involves catching of errors generated by program followed by taking of correct action. If you would handle errors appropriately then it may lead to many unexpected consequences. Error handling in PHP is very easy.

Few techniques related to Process of error handling in PHP are mentioned below :

- using die() function
- using set_error_handler()
- using trigger_error()

using die() function :

Consider, the following sample code of opening a file named "sample.txt", If the file "sample.txt" is not present at current location then it will give error as explained in below mentioned sample code

\

Sample code 4.3.a

```
<?php

    $file=fopen("sample.txt","r"); //file open in read mode

?>
```

Output :

Warning: fopen(sample.txt): failed to open stream: No such file or directory in /opt/lampp/htdocs/er.php on line 3

Explanation :

It gives warning as file is not exist at the current location.

Solution for the above mentioned code is ,With the help of die() function we can solve this problem as explained in below mentioned sample code

Sample Code 4.3.b

```
<?php

if(file_exists("sample.txt")) {

    $file = fopen("sample.txt", "r");

}

else {

    die("Error_File does not exist");
```

```
}
```

```
?>
```

Output :

Error_File does not exist

This is one of the simplest way to handle errors

Using set_error_handler() :

Consider Following sample code

Sample Code 4.3.c

```
<?php
```

```
$x=10;
```

```
$y=20;
```

```
echo($z);
```

```
?>
```

Output :

Notice: Undefined variable: z in /opt/lampp/htdocs/er.php on line 7

Explanation :

Above mentioned sample code generate Notice instead of error

Solution :

using `set_error_handler()` function we can generate proper error as shown in below mentioned

Sample Code 4.3.d

```
<?php

function add($errno, $errstr) {

    echo "Error:[$errno] $errstr";

}

set_error_handler("add");

$x=10;

$y=20;

echo($z);

?>
```

Output :

Error:[8] Undefined variable: z

Using `trigger_error()` :

using `trigger_error()` function, it is possible to triggered error anywhere in the program where we wish to triggered.

Consider the below mentioned sample code

Sample Code 4.3.e

```
<?php

$x=10;

$y=2.5;

$z=$x/$y;

if ($y>=2){

    trigger_error("Value of y must be less than 2");

}

?>
```

Output :

Notice: Value of y must be less than 2 in **/opt/lampp/htdocs/er.php** on line **9**

Explanation :

As mentioned in the above sample code if we want that value of denominator must not be greater than zero then we can triggered the error that Value of y must be less than 2.

Addition to above mentioned methods second parameter could be added to specify an error level. Consider below mentioned sample codes

Use of E_USER_WARNING as a second parameter :

Source Code 4.3.f

```
<?php

function div($errno, $errstr) {

    echo "Error:[$errno] $errstr";

}

set_error_handler("div",E_USER_WARNING);


$x=10;

$y=2.5;

$z=$x/$y;

if($y>=2) {

    trigger_error("Value of y must be less than 2",E_USER_WARNING);

}

?>
```

Output :

Error:[512] Value of y must be less than 2

Use of E_USER_ERROR as a second parameter :

Sample Code 4.3.g

```
<?php

function div($errno, $errstr) {

    echo "Error:[$errno] $errstr";

}

set_error_handler("div",E_USER_ERROR);

$x=10;

$y=2.5;

$z=$x/$y;

if($y>=2) {

    trigger_error("Value of y must be less than 2",E_USER_ERROR);

}

?>
```

Output :

Error:[256] Value of y must be less than 2

Use of E_USER_NOTICE as a second parameter :

Sample Code 4.3.h

```
<?php

function div($errno, $errstr) {
```

```

        echo "Error:[$errno] $errstr";

    }

    set_error_handler("div",E_USER_NOTICE);

    $x=10;

    $y=2.5;

    $z=$x/$y;

    if($y>=2) {

        trigger_error("Value of y must be less than 2",E_USER_NOTICE);

    }

?>

```

Output :

Error:[1024] Value of y must be less than 2

Self Test (Multiple Choice Questions):

1. What will be the output of the following code

```

<?php

$x=5;

$y=10;

if($x>10) {

    $z=$x+$y;

```

```
        echo $z;

    }

    else {

        die("Value x is less than 10 ");

    }

?>
```

Options :

a.15

b.Value x is less than 10

c.10

d.None of above

2.What will be the output of following code

```
<?php

$x=;

$y=20;

echo $x;

?>
```

Options :

a.Parse error: syntax error, unexpected ';' in /opt/lampp/htdocs/er.php on line 3

b.Error:[8] Undefined variable: x

c.both a&b

d.None of above

3.What will be the output of following code

```
<?php
```

```
function ($errno, $errstr) {
```

```
    echo "Error:[$errno] $errstr";
```

```
}
```

```
set_error_handler("add",E_USER_ERROR);
```

```
$x=10;
```

```
$y=3.5;
```

```
$z=$x+$y;
```

```
if($y>=3) {
```

```
    trigger_error("Value of y must be less than 3",E_USER_ERROR);
```

```
}
```

```
?>
```

Options :

a.Value of y must be less than 3

b.Error:[256] Value of y must be less than 3

c.**Parse error:** syntax error, unexpected 'set_error_handler' (T_STRING) in /opt/lampp/htdocs/er.php on line 7

d.both c&d

4. using _____ function it is possible to triggered error anywhere in the program

a.trigger_error()

b. set_error_handler()

c.die()

d.both a&b

5. trigger_error() have at least _____ no of parameter

a.1 b. 2 c.3 d.None of above

4.4 Parse error : Syntax Error

There are various types of errors and warnings generated after the execution of program. Parse errors generated dues to missing or Extra parentheses, extra or braces are unclosed,semicolon is Missing ,quotes are unclosed etc.

Parse error or syntax error :

Parse error or syntax error can generate if the syntax mistake happened. Consider we have php code stored in sample.php file for this file Parse error: syntax error can be anything like unexpected ‘{’ in sample.php or expecting ‘;’ in sample.php on line 55

Sample Code 4.4.a

```
<?php
```

```
class Subject {
```

```
    public $name;
```

```

function Input($name) {

    $this->name = $name;

}

function Show() {

    return $this->name;

}

}

$PHP=new Subject

$PHP->Input('PHP');

echo $PHP->Show();

?>

```

Output :

Parse error: syntax error, unexpected '\$PHP' (T_VARIABLE) in **/opt/lampp/htdocs/hi.php** on line **18**

Explanation:

In above mentioned code 4.4.a Semicolon is missing on line 18,it gives **Parse error:** syntax error. It does not refer to a quoted "VARIABLE". It means a raw identifier was encountered.

Sample Code 4.4.b

```

<?php

$x = 10;

```

```
y = 20;
```

```
$z=$x+$y;
```

```
echo $z;
```

```
?>
```

Output:

Parse error: syntax error, unexpected '=' in /opt/lampp/htdocs/hi.php on line 3

Explanation :

In above code 4.4.b **Parse error:** syntax error, unexpected '=' in /opt/lampp/htdocs/hi.php on line 3 occurs because **\$ sign is missing in line 3**

We need to solve these errors in different approaches like more regularly look at preceding lines or comment out the code which causing problems,etc.

Self Test (Multiple Choice Questions):

1. What will be the output of following code

```
<?php

class Subject {

    public $name;

    function Input($name) {

        $this->name = $name;

    }

    function Show() {

        return $this->name;

    }

}

$PHP=new Subject;

$PHP->Input('php');

echo $PHP->Show();

?>
```

Options:

a.**Parse error:** syntax error, unexpected '\$PHP' (T_VARIABLE) in /opt/lampp/htdocs/er.php on line 19

b. php

c.name

d. None of above

2. What will be the output of following code

```
<?php
```

```
$x = 10; $y = 20;
```

```
z=$x+$y;
```

```
echo $z;
```

```
?>
```

Options :

a. **Parse error: syntax error, unexpected '=' in /opt/lampp/htdocs/er.php on line 5**

b. **Parse error: syntax error, undefined z in /opt/lampp/htdocs/er.php on line 5**

c. 30

d. None of above

3. unexpected '{' is a Syntax Error

a. True b. False

4.5 Undefined index

At the time of execution of PHP code you may get an error undefined index.

Sample Code 4.5.a

```
<?php
```

```
$name = $_GET['name'];
```

```
$address = $_GET['address'];
```

```
echo $name;
```

```
echo $address;
```

```
?>
```

Output:

Notice: Undefined index: name in /opt/lampp/htdocs/hi.php on line 4

Notice: Undefined index: address in /opt/lampp/htdocs/hi.php on line 5

Explanation :

In sample code 4.5.a values are not assigned to variable \$name and \$ address therefore it gives Notice of Undefined index

This error can be solved using isset () function as shown in sample code 4.5.b

Sample Code 4.5.b

```
<?php
```

```
if(isset($_GET['name'])){
```

```
    $name = $_GET['name'];
```

```
}
```

```
else{
```

```
    $name = "Name not set";
```

```
}
```

```
if(isset($_GET['address'])){
```

```
    $address = $_GET['address'];
```

```
}
```

```
else{
```

```
    $address = "<br>Address not set ";
```

```
}
```

```
echo $name;
```

```
echo $address;
```

```
?>
```

Output :

Name not set

Address not set

4.6 Error Reporting :

To specify which type of error is reported `error_reporting()` function is used. Errors are having many levels using `error_reporting()` function one can set the level of reporting.

For example,

`error_reporting(0)` indicates that all error reporting are turn off.

`error_reporting(E_ERROR | E_WARNING | E_PARSE)` indicates that simple errors are running.

4.7 Exception Handling

With the help of exception handling we can change the flow of the execution of program in case of exception condition occurs.

Like other object oriented programming language exception handling in PHP is very simple PHP also provides different keywords to handle exception in PHP as mentioned below

try : It is a block of code where we can code for which exception can occur

catch : If in case any exception is thrown catch block is responsible to execute that particular exception

throw : This block is use to throw exception

finally : This block is use after the catch block

Sample Code 4.7.a

```
<?php
function Sample($value) {
    try {
        if($value == 5) {

            throw new Exception('Number is 5 </br>');

        }
    }
}
```

```

    }
    catch(Exception $e) {

        echo "</br>Exception Caught properly ", $e->getMessage();

    }

    echo "</br>This statement after catch will be always executed";
}
Sample(55);
Sample(5);
?>

```

Output :

This statement after catch will be always executed
 Exception Caught properly Number is 5

This statement after catch will be always executed

Sample Code 4.7.b

```

<?php
function Sample($value) {
    try {
        if($value == 5) {
            throw new Exception('Number is 5 </br>');
        }
    }
    catch(Exception $e) {
        echo "</br>Exception Caught properly ", $e->getMessage();
    }
    finally {
        echo "This block clean all the activity ";
    }
}
Sample(55);
Sample(5);

```

?>

Output :

This block clean all the activity

Exception Caught properly Number is 5

This block clean all the activity

Use of Exceptions

When any exception is thrown, the code next to it will not be executed, and PHP will try to matching "catch" block. If an exception is not caught then the fatal error will be issued with an "Uncaught Exception" message.

Consider an example that try to throw an exception without catching it:

Sample Code 4.7.c

```
<?php

function checkNumber($num){

    if($num>1)

        throw new Exception("number must be less than or equal to 1");

    return true;

}

checkNumber(2);

?>
```

Output

Fatal error: Uncaught Exception: number must be less than or equal to 1 in /opt/lampp/htdocs/er.php:5 Stack trace: #0 /opt/lampp/htdocs/er.php(10): checkNumber(2) #1 {main} thrown in /opt/lampp/htdocs/er.php on line 5

Exception handling program works in below mentioned flow

Step 1 : Program code is checked whether the exception occurred or not

Step 2 :

2.A : If exception occurred then check exception handled or not

2.A.a : If exception is handled Finally block is executed

2.A.b : If exception is not handled then also Finally block is executed

2.B:If exception not occurred

2.B.a: If exception not occurred then Finally block is executed.

Self Test (Multiple Choice Questions):

1. What will be the output of following code

```
<?php
```

```
function Test($value) {
```

```
    try {
```

```
        if($value == 9) {
```

```
            throw new Exception('Number is 9 </br>');
```

```
        }
```

```
    }
```

```
    catch(Exception $e) {
```

```

        echo "</br>Exception Caught properly ", $e->getMessage();

    }

    finally {

        echo "This block clean all the activity ";

    }

}

Test(9);

?>

```

Options :

- a.This block clean all the activity
Exception Caught properly Number is 9
This block clean all the activity
- b.Exception Caught properly Number is 9
This block clean all the activity
- c.Number is 9
- d.This block clean all the activity

2. What will be the output of following code

```

<?php

function Test($value) {

    try {

```

```

        if($value == 9) {

            throw new Exception('Number is 9 </br>');

        }

    }

    catch(Exception $e) {

        echo "</br>Exception Caught properly ";

    }

}

Test(99);

Test(9);

?>

```

Options :

- a.Exception Caught properly Number is 9
This block clean all the activity
- b.Exception Caught properly
- c.Exception Caught properly Number is 9
- d.None of above

3._____ It is a block of code where we can write code for which exception can occur

Options

- a.try
- b.catch

c.finally

d. None of above

4. _____ block is use after the catch block

a.try

b.catch

c.finally

d. None of above

5. _____ block is use to throw exception

a.try

b.catch

c.finally

d. None of above

4.8 Summary

With the help of this unit students can understand types of errors and exceptions and also students understand that how to handle Errors as well as Exception occurred at the time of execution of source code.

4.9 Exercise:

1. Why Exception Handling in PHP?
2. State and explain various PHP specialized keywords.
3. What are the main error types in PHP and how do they differ?
4. How can you enable error reporting in PHP?
5. How does one prevent the following Warning 'Warning: Cannot modify header information – headers already sent' and why does it occur in the first place?
6. List and Describe PHP error constants.
7. What will be the output/error of following code?

```
<?php
    $x = "xyz";
    y = "pqr";
    echo $x;
    echo $y;
?>
```

8. Explain Basic Error Handling die() function. What will be the output of following code?

```
<?php
    if(!file_exists("welcome.txt"))
    {
        die("File not found");
    }
    Else
    {
        $file=fopen("welcome.txt","r");
    }
?>
```

9. State & explain various rules for exceptions.
10. How to create Custom Error Handler? Explain error_function() with parameters in detail.

Unit 5

PHP MySQLi

5.1 Learning Objectives

After successful completion of this unit, Students will be able to perform database connectivity with PHP.

5.2 Introduction

This chapter focuses on **MySQLi** or **MySQL extension** with PHP. To access MySQL database servers MySQLi functions are used. we can use MySQLi extension with MySQL version 4.1.13 or newer.

5.3 MySQLi connect

We can use both object-oriented and procedural interface for accessing mysql database using MySQLi.

Using Procedural approach :

Sample Code 5.3.a

```
<?php
$cn = mysqli_connect("localhost","",""); // Establish connection

if (!$cn) { // Connection Checking

    die("Connection is not done properly: " . mysqli_connect_error());
}
echo "Connection established successfully";
?>
```

Output :

Connection established successfully

Explanation :

In above mentioned sample code 5.3.a is having procedural approach to establish connection with mysql and `mysqli_connect` function is used to establish connection with mysql

`mysqli_connect()` is having five parameters mentioned below

- host : Name of host
- username : Username to access the database
- password : Password provided to the database,
- dbname : Name of Database
- port : Port Number
- socket : Socket

`mysqli_connect(host, username, password, dbname, port, socket)`

In sample code 5.3.a only three parameters mentioned(host, username, password) in `mysqli_connect` whereas port and socket are optional parameter

In sample code 5.3.a

```
$cn = mysqli_connect("localhost","", ""); // Establish connection
```

\$cn is used to stored the connection string from `mysqli_connect`

Name of host : localhost

If username and password are not set then we can mentioned as empty ("") as shown in sample code 5.3.a

Using object-oriented approach :**Sample Code 5.3.b :**

```
<?php
```

```
$cn = new mysqli("localhost", "", "");
```

```
if (!$cn) { // Connection Checking
```

```
    die(" Connection is not done properly : " . mysqli_connect_error());
```

```
}
```

```
echo "Connection established successfully";
```

```
?>
```

Explanation :

In Sample Code 5.3.b instance of mysqli class is created using new. Once instance created successfully then we can say that the connection is successful.

All required details are provided with mysqli class to create connection like Name of host - localhost. If username and password are not set then we can mentioned as empty ("").

Self Test (Multiple Choice Questions) :

1) Which of the following is an optional parameter:

Options:

- a.host
- b.dbname
- c.username
- d.socket

2) Which of the following is not a parameter of mysqli_connect():

Options:

- a.host
- b.username
- c.port
- d.query

3) To work with mysqli which class is instantiated via its constructor:

Options:

- a.mysql
- b.msqli
- c.sql
- d.sqli

4) Which of the following is correct syntax when the username and password is not set?

Options:

- a. \$cn = new mysqli("localhost", "", "");
- b. \$cn = new mysqli("localhost", "NA", "NA");
- c. \$cn = new mysqli("localhost", "_", "_");
- d. \$cn = new mysqli("localhost", " ", " ");

5.4 Loop through MySQLi results

The MySQLi extension can also manage result sets object using the `fetch_array()`, `fetch_row()`, `fetch_object()` and `fetch_array()` methods, respectively. Their prototypes follow:

It is very easy to get data from database using `mysqli` function inside loop like `while` statement. If in case loop gets fails to fetch the next row, it returns false, and loop get ends.

Sample Code 5.4.a :

```
<?php
$cn = mysqli_connect("localhost","root","", "mysql"); // Establish connection
if (!$cn) { // Connection Checking

    die("Connection is not done properly: " . mysqli_connect_error());
}
$res=mysqli_query($cn,"SELECT * FROM stud");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Address</th>
<th>Mobile number</th>
</tr>";
```

```

while($row = mysqli_fetch_array($res))
{
    echo "<tr>";
    echo "<td>" . $row['fname'] . "</td>";
    echo "<td>" . $row['lname'] . "</td>";
    echo "<td>" . $row['addd'] . "</td>";
    echo "<td>" . $row['mnumber'] . "</td>";
    echo "</tr>";
}
echo "</table>";
mysqli_close($cn); //close the connection
?>

```

Output :

Firstname	Lastname	Address	Mobile number
Prashant	Patil	Nashik	0
Vivek	Patil	Nashik	0
Ritesh	Suryawanshi	Pune	0

Explanation :

In Sample Code 5.4.a `mysqli_query()` function is having two parameters, first is connection string stored in `$cn` and second is query which will executes on `$cn`. And the result of `mysqli_query()` is stored in `$res`.

Here in Sample Code 5.4.a we have used `mysqli_fetch_array()` function inside while loop to retrieve the data in rows from database.

Self Test (Multiple Choice Questions) :

1) Which of the following is not a method of the Mysqli extension?

Options:

- a. fetch_array()
- b. fetch_row()
- c. fetch_object()
- d. fetch_data()

2) Is it possible to loop through Mysqli results?

Options:

- a. Yes
- b. No
- c. Cannot say

3) Which of the following statement is used to loop through Mysqli results?

Options:

- a. if
- b. while
- c. switch
- d. None of the above

4) Which of the following method is used inside while loop to retrieve the data in rows from the database?

Options:

- a. mysqli_query()
- b. fetch_data()
- c. mysqli_fetch_array()
- d. mysqli_debug()

5.5 Prepared statements in MySQLi

MySQLi provides is one of the best feature named prepare statement, with the use of prepare statement we can execute similar SQL statements repeatedly.

Prepare statements are highly efficient as the parsing time of prepare statement is very less since the preparation for the query is done once and we can executes statements multiple times.

Sample Code 5.5.a

```
<?php

$cn = mysqli_connect("localhost","root","","mysql"); // Establish connection

if (!$cn) {                                // Connection Checking

    die("Connection is not done properly: " . mysqli_connect_error());

}

$st = $cn->prepare("INSERT INTO stud(fname,lname,addd,mnumber) VALUES (?, ?, ?,?)");

$st->bind_param("sssi",$fname,$lname,$addd,$mnumber);

$fname = "Pushkaraj";
$lname = "Kasture";
$addd = "Pushkaraj@example.com";
$mnumber=0000000000;

$st->execute();

$fname = "Rahul";
$lname = "Patil";
$addd = "Rahul@example.com";
$mnumber=0000000000;

$st->execute();
```

```
$fname = "Atul";  
$lname = "Chaudhari";  
$add = "Atul@example.com";  
$mnumber=0000000000;  
$st->execute();  
  
echo "Records generated successfully";  
  
$st->close();  
$cn->close();  
  
?>
```

Output :

Records generated successfully

Explanation :

In above code 5.5.a `bind_param()` function is responsible to binds the parameters with the SQL query and at the same time it specifies argument types. Types of argument can be one of following types

- s indicates string
- d indicates double
- i indicates integer
- b indicates BLOB

Self Test (Multiple Choice Questions) :

1)With the use of _____ statement can the user execute similar SQL statements repeatedly

Option :

- a.prepare
- b.view
- c.stored procedure
- d.template

2)Which if the following are a type of argument of bind_param()

- i. s indicates string
- ii. c indicates char
- iii. i indicates integer
- iv. b indicates BLOB

Option :

- a.i, ii, iii
- b.i, iii, iv
- c.i, ii, iv
- d.i, iii

3)Prepare statement is not supported in Mysqli?

Option :

- a.Yes
- b.No

4)Which of the following does not support the SQL queries with *?

Option :

- a.bind_result()
- b.get_result()
- c.store_result()
- d.None Of the Above

5.6 Escaping Strings :

To escapes special characters from string to use in an SQL query `mysqli_real_escape_string()` function is used

Syntax of `mysqli_real_escape_string()` function :

`mysqli_real_escape_string(connection, escapestring)`

Sample Code 5.6.a :

```
<html>

<head>

<title>Form Input Data</title>

</head>


<body>

<table border="1">

  <tr>

    <td align="center">Form Input Student Data</td>

  </tr>

  <tr>

    <td>

      <table>

        <form method="post" action="">

          <tr>

            <td>First Name</td>

            <td><input type="text" name="fname" size="20">

          </td>

        </tr>

        <tr>

            <td>Last Name</td>

            <td><input type="text" name="lname" size="40">

          </td>

        </tr>

      </table>

    </td>

  </tr>

</table>
```

```

        <tr>
        <td>Address</td>
        <td><input type="text" name="addd" size="40">
        </td>
        </tr>

        <tr>
        <td>Mobile Number</td>
        <td><input type="text" name="mnumber" size="40">
        </td>
        </tr>

        <tr>
        <td></td>
        <td align="right"><input type="submit" name="submit" value="Sent"></td>
        </tr>
    </table>
</td>
</tr>
</table>
</body>
</html>

```

```
<?php
```

```

$cn=mysqli_connect("localhost","root","","mysql");// Establish connection
if (!$cn) // Connection Checking
{
    die("Connection failed: " . mysqli_connect_error());
}

// Escape special characters, if any
$fname = mysqli_real_escape_string($cn,$_POST['fname']);

```

```

$name = mysqli_real_escape_string($cn,$_POST['lname']);
$add = mysqli_real_escape_string($cn,$_POST['add']);
$mnumber = mysqli_real_escape_string($cn,$_POST['mnumber']);
$sql="INSERT INTO stud(fname,lname,add,mnumber) VALUES
('$fname','$lname','$add','$mnumber)";
mysqli_query($cn, $sql);
mysqli_close($cn); //close the connection
?>

```

Output :

Input Student Data	
First Name	<input type="text" value="Nilesh"/>
Last Name	<input type="text" value="Patil"/>
Address	<input type="text" value="Nashik"/>
Mobile Number	<input type="text" value="000000000"/>
<input type="button" value="Sent"/>	

Explanation :

In above Sample Code 5.6.a we have escaped the special characters from the inputs taken from HTML file .

Escapes special characters and Data get inserted.

Self Test (Multiple Choice Questions) :

1) Which of the following is correct Syntax of escaping strings in mysqli?

Option :

- a. `mysqli_real_escape_string(conn, escapestrings)`
- b. `mysqli_real_escape_string(conn, escapestrings())`
- c. `mysqli_real_escape_string(conn, escapestrings{ })`
- d. `mysqli_real_escape_string(conn, escapestrings[])`

2) While escaping strings Data is also escaped with the special characters

Option :

a. Yes

b. No

5.7 Debugging SQL in MySQLi :

Debugging operations is performed with the help of **mysqli_debug()** function.

mysqli_debug() function requires parameter as a string representation to perform debugging operation.

5.8 MySQLi query :

MySQL Create Table :

Consider the example of Sample Code 5.8.a where we are creating the table into the database with name staff using php

Sample Code 5.8.a:

```
<?php
$cn = new mysqli("localhost", "root", "", "mysql");
if (!$cn) {      // Connection Checking
    die("Connection failed: " . mysqli_connect_error());
}
$sql = "create table staff(id INT,name VARCHAR(20))";

if(mysqli_query($cn,$sql)) {
    echo "Table staff created successfully";
}
else{
echo "Could not create table: ". mysqli_error($cn);
}
mysqli_close($cn); //close the connection
?>
```

Output:

Table staff created successfully.

Explanation :

In Sample code 5.8.a we have created table in Mysql using create table query of mysql and executed on connection using `mysqli_query()` function.

MySQL Insert Record :

Consider Staff table is available with the database and we have to insert values of id and name into the table through php.

Consider following example of Sample Code 5.8.b

Sample Code 5.8.b

```
<?php
$cn = new mysqli("localhost", "root", "", "mysql");
if (!$cn) {                                // Connection Checking

    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO staff(id,name) VALUES (5,'Bhushan)";
if(mysqli_query($cn, $sql)){
    echo "Record successfully inserted into staff table";
}
else{
    echo "Could not insert record: " . mysqli_error($cn);
}

mysqli_close($cn); //close the connection
?>
```

Output :

Record successfully inserted into staff table

Explanation :

In Sample code 5.8.b we have inserted record in staff table in Mysql using insert query of mysql and executed on connection using `mysqli_query()` function.

If we want to enter the data into the database from web page for that reason we have to perform the following steps :

1. Design web page in HTML
2. Pass the values from HTML to PHP code using post method
3. Insert the values into the database through insert query

Below mentioned is the sample code 5.8.c

Sample code 5.8.c

```
<html>
<head>
<title>Form Input Data</title>
</head>

<body>
<table border="1">
  <tr>
    <td align="center">Form Input Student Data</td>
  </tr>
  <tr>
    <td>
      <table>
        <form method="post" action="">
          <tr>
            <td>First Name</td>
            <td><input type="text" name="fname" size="20">
          </td>
```

```

</tr>

<tr>

    <td>Last Name</td>

    <td><input type="text" name="lname" size="40">

    </td>

</tr>

    <tr>

    <td>Address</td>

    <td><input type="text" name="add" size="40">

    </td>

</tr>

    <tr>

    <td>Mobile Number</td>

    <td><input type="text" name="mnumber" size="40">

    </td>

</tr>

<tr>

    <td></td>

    <td align="right"><input type="submit" name="submit" value="Sent"></td>

</tr>

</table>

</td>

</tr>

</table>

</body>

</html>

```

```

<?php

```

```

$cn =mysqli_connect("localhost","root","","mysql"); // Establish connection
if (!$cn) // Connection Checking

```

```

{
    die("Connection failed: " . mysqli_connect_error());
}

// Escape special characters, if any
$fname = $_POST['fname'];
$lname = $_POST['lname'];
$add = $_POST['add'];
$mnumber = $_POST['mnumber'];

$sql="INSERT INTO stud(fname,lname,add,mnumber) VALUES
('$fname','$lname','$add','$mnumber)";
mysqli_query($cn, $sql);
mysqli_close($cn); //close the connection
?>

```

MySQL Update Record :

Consider example mentioned in Sample Code 5.8.d where updation of record is done through update query

Sample Code 5.8.d

```

<?php
$cn = new mysqli("localhost", "root", "", "mysql");

if (!$cn) {
    // Connection Checking
    die("Connection failed: " . mysqli_connect_error());
}

$sql="update staff set name='Kuldeep' where id=5";

if(mysqli_query($cn,$sql)) {
    echo "Name updated successfully";
}

```

```

}
else{
    echo "Could not update record: ". mysqli_error($cn);
}
mysqli_close($cn); //close the connection
?>

```

Output :

Name updated successfully

Explanation :

In Sample code 5.8.d we have updated record in staff table in Mysql using update query of mysql and executed on connection using mysqli_query() function.

MySQL Delete Record :

Consider example of Sample Code 5.6.d where we are deleting the values from the database using php code.

Sample Code 5.8.e

```

<?php
$cn = new mysqli("localhost", "root", "", "mysql");
if (!$cn) {
    // Connection Checking
    die("Connection failed: " . mysqli_connect_error());
}
$sql = "delete from staff where id=5";
if(mysqli_query($cn, $sql)) {
    echo "Row deleted successfully";
}
else {
    echo "Could not deleted record: ". mysqli_error($cn);
}
mysqli_close($cn);

```

?>

Output :

Row deleted successfully

Explanation :

In Sample code 5.8.e we have deleted record in staff table in Mysql using update delete of mysql and executed on connection using mysqli_query() function.

MySQLi Select Query Example :

Please refer code 5.4.a for Select Query Example

we have selected record from table in Mysql using select query of mysql and then fetched rows using function mysqli_fetch_array() function inside while loop to retrieve the data in rows from database.

5.9 How to get data from a prepared statement :

For Explanation refer section 5.5.a

5.10 MySQLi Insert ID :

mysqli_insert_id() function used for MySQLi Insert ID which returns the id

Sample Code 5.10.a :

```
<?php
```

```
$cn = mysqli_connect("localhost","root","","mysql"); // Establish connection
```

```
if (!$cn) { // Connection Checking
```

```
    die("Connection failed: " . mysqli_connect_error());
```

```
}
```

```
$sql = "INSERT INTO employee(name,addd) VALUES ('Tushar','Pune')";
```

```
mysqli_query($cn, $sql);
```

```
echo "New record has id: " . mysqli_insert_id($cn);
```

```
mysqli_close($cn);
```

?>

Output :

New record has id: 0

5.11 Close connection

mysqli_close() function use to close previously opened database connection refer code 5.10.a where connection established with connection string \$cn is close.

5.12 Joins :

SQL JOIN :

JOIN is used to combine records from two or more tables.

Following are the different types of Joins :

1. Inner Join : Returns the records matching in both tables
2. Left Join : Returns all records from left table matched records from right table.
3. Right Join : Returns all records from Right table matched records from left table.
4. Full Join : Returns all records from both left table and right table matched records from both the tables.

5.13 Summary

With the help of this unit students can perform php and mysql connectivity, understand and working meanig of funcitons required for database connectivity

5.14 Exercise :

1. How can we connect to a MySQL database from a PHP script?
2. Explain How is it possible to know the number of rows returned in the result set?
3. How can we create a database using PHP and MySQL?
4. Explain the use of `mysqli_debug()` function.
5. Explain the benefits of Prepared statements in MySQLi
6. Explain the syntactical difference between procedural and object-oriented approach while defining Mysqli functions.
7. Explain the use of `mysqli_real_escape_string` function with example.
8. Explain the use of `mysqli_fetch_array` with sample code.
9. `mysqli_debug()` function requires a parameter as a _____ representation to perform debugging operation

Options :

- a.string
- b.float
- c.int
- d.double

10. Which of the following works with all SQL statements?

Options :

- a. with `bind_result`, and then fetch on the statement object
- b. with `get_result`, and then `fetch_assoc` (or other `fetch_*` variant) on the result object
- c. Both
- d. None of the Above

11. What does `mysqli_insert_id()` return?

Options :

- a. auto generated ID in the latest query

b.string value of the query

c.first ID of the column

d.average of the ID's

12. mysqli_insert_id() return string when _____.

Options :

a.number is greater than int value

b.number is equal to the int value

c.number is less than the int value

d.number is negative

13. JOIN is used to combine records of 2 or more tables?

Options :

a.True

b.False

14. Which of the following is not a type of JOIN?

Options :

a.Inner Join

b.Left Join

c.Partial Join

d.Full Join

Unit 6

Object Oriented Programming

6.1 Learning Objectives

After successful completion of this unit, you will be able to developed object oriented functionality in PHP.

6.2 Introduction

In this chapter, we will study the various object oriented concepts such as Class, Object, Member Function, Constructor, Destructor, Inheritance, Function Overloading, Access Specifier, etc. used in PHP programming.

If we compare object oriented programming with procedural oriented programming, object oriented programming is much faster. And also with the help of object oriented programming we can reuse the code it will help use avoid repetition of code.

Class and Object are two major part of object oriented programming. Class consist of member variable,member function whereas object is an instance of class.

6.3 Defining PHP Classes

Class is defined using class keyword followed by the classname followed by curly braces. Below mentioned is the syntax of class

```
<?php
```

```
class Student {
```

```
}
```

```
?>
```

6.4 Creating Objects in PHP

new keyword is required to create Object of a class as shown in Sample Code 6.4.a.

One class may have multiple objects as shown in Sample Code 6.4.b. If we create multiple objects of a class then every object is having all the properties as well as methods define inside the class with different property values.

Sample Code 6.4.a

```
<?php

class Student {

    public $name;

    function Input($name) {

        $this->name = $name;

    }

    function Display() {

        return $this->name;

    }

}

$obj=new Student;

echo $obj->Input("Vivek");
```

```
echo $obj->Display();
```

```
?>
```

Output:

Vivek

Explanation :

Object \$obj is created by using new keyword where Student is name of class. We are passing the string as a parameter to Input method of class by calling Input method through object \$obj and displaying the value of string using the same object.

Sample Code 6.4.b

```
<?php
```

```
class Student{
```

```
    public $name;
```

```
    function Input($name) {
```

```
        $this->name = $name;
```

```
    }
```

```
    function Display() {
```

```
        return $this->name;
```

```
    }
```

```
}
```

```
$obj=new Student;
```

```
echo $obj->Input("Vivek");
```

```
echo $obj->Display();
```

```
$obj2=new Student;
```

```
echo $obj2->Input("Patil");
```

```
echo $obj2->Display();
```

```
?>
```

Output :

VivekPatil

Explanation :

Object \$obj and \$obj2 are created of same class Student using new keyword.

We are passing the string “Vivek ”as a parameter to Input method of class by calling Input method through object \$obj and displaying the value of string using \$obj.

Similarly, We are passing the string “Patil ”as a parameter to Input method of class by calling Input method through object \$obj2 and displaying the value of string using \$obj2.

Creating Multiple Classes and Multiple Objects:

Below is the sample code of 6.4.c contains multiple classes and multiple objects

Sample Code 6.4.c

```
<?php
```

```
class Student{
```

```
    public $name;
```

```
        function Input($name) {
```

```
            $this->name = $name;
```

```
        }
```

```
function Display() {
```

```
    return $this->name;
```

```
}
```

```
}
```

```
class Staff{
```

```
    public $name;
```

```
        function Input($name) {
```

```
            $this->name = $name;
```

```
        }
```

```
function Display() {  
  
    return $this->name;  
  
}  
  
}  
  
$stud_obj=new Student;  
  
echo $stud_obj->Input("Sagar");  
  
echo $stud_obj->Display();  
  
$stud_obj2=new Student;  
  
echo $stud_obj2->Input("Bhosale <br>");  
  
echo $stud_obj2->Display();  
  
$staff_obj=new Student;  
  
echo $staff_obj->Input("Mohan");  
  
echo $staff_obj->Display();  
  
$staff_obj2=new Student;  
  
echo $staff_obj2->Input("Nikam");  
  
echo $staff_obj2->Display();
```

?>

Output :

SagarBhosale

MohanNikam

Self Test (Multiple Choice Questions)

1. Which Keyword is used to create Object of class in PHP?

Options :

- a. New
- b. Public
- c. Object
- d. None of above

2. How many objects can be created for a class?

Options :

- a. 2
- b. 1
- c. Multiple
- d. Maximum 4

3. Correct Syntax to create a objectis -

Options :

- a. \$variable=class_name();
- b. \$variable=new class_name();

c. \$variable=new class class_name();

d. new.\$variable=class_name();

4. Object is an _____ of a class.

Options :

a. Instance

b. Prototype

c. a and b

d. Function

5. What will be the output of the code

```
<?php
```

```
class Student {
```

```
    public $name;
```

```
    function Input() {
```

```
        $this->name = "php";
```

```
    }
```

```
    function Display() {
```

```
        return $this->name;
```

```
    }
```

```
}
```

```
$obj=new Student;
```



```
echo $obj->Input();
```

```
echo $obj->Display();
```

```
?>
```

Options:

a.php

b.name

c.\$this→name

d.None of above

6.5 Calling Member Functions

With the help of object of a class we can call member function of that particular class. We Can also call the same member functions with different properties by creating multiple objects of the class.

Consider example of Sample Code 6.4.b, In Sample Code 6.4.b Input and Display methods called by object name \$obj and similarly same methods are called using \$obj2.

Self Test (Multiple Choice Questions)

1. What will be the Output of following?

```
<?php
```

```
class hello{
```

```
function world() {
```

```
echo"Hello world";
```

```
}  
  
}  
  
$a=new hello();  
  
$a->world();  
  
?>
```

Options :

- a. Error
- b. True
- c. No output
- d. Hello World

2. We can call member function using Object of particular class with different properties.

Options :

- a. True
- b. False

3. Member function can access member's of_____

Options:

- a. Current object
- b. All objects
- c. Both a and b
- d. None of these

6.6 Constructor Functions

Like other object oriented programming language Constructor in PHP is also used to initialize properties of an object and also like other object oriented programming language Constructor Functions are automatically called at the time of constructing an object.

In PHP also constructor is a special function. `__construct()` defines the constructor.

Sample Code 6.6.a

```
<?php

class Student {

    public $name;

    function __construct() { // Define Constructor

        $this->name="Vivek"; //name get initialize in constructor

    }

    function Display() {

        return $this->name;

    }

}

$obj = new Student; // Constructor called

echo $obj->Display();

?>
```

Output :

Vivek

Explanation :

In Sample Code 6.6.a constructor is called automatically when object \$obj is created and name get initialize in constructor.

Sample Code 6.6.b

```
<?php

class Student {

    public $name;

    function __construct($name) { // Define Constructor

        $this->name=$name; //name get initialize in constructor

    }

    function Display() {

        return $this->name;

    }

}

// Constructor called

$obj= new Student("Sagar");

echo $obj->Display();

?>
```

Output :

Sagar

Self Test (Multiple Choice Questions)

1. Constructor Functions are automatically called at the time of

Options:

- a. Calling of method
- b. constructing an object.
- c. Defining class
- d. Defining method

2. In php which function used to define constructor?

Options:

- a. Construcor()
- b. __construct()
- c. __ constructor()
- d. Construct()

3. Constructor also called as _____ in PHP

Options:

- a. Magic functions
- b. Static member of class
- c. Object of class
- d. All of above

4. Constructor have

Options:

- a. parameters
- b. No parameters
- c. Only one arguments
- d. a or b

6.7 Destructor

`__destruct()` function is used to define destructor in PHP. This function automatically called when the script stopped. Syntax of destructor is having two underscores before destruct word.

Sample Code 6.7.a

```
<?php

class Student {

    public $name;

    function __construct() { // Define Constructor

        $this->name="Vivek"; //name get initialize in constructor

    }

    function __destruct() { //Define Destructor

        echo $this->name; // Print the value of name inside Destructor

    }

}
```

```
}
```

```
$obj = new Student; // Constructor called
```

```
?>
```

Output :

Vivek

Explanation :

__destruct() get called at the end of the script.

➤ **Self Test (Multiple Choice Questions)**

1. Destructor is called_____

Options :

- a. When destructor defined
- b. When object is destructed or the script is stopped.
- c. Not called if not defined
- d. When constructor passess control to destructor

2. In php which function used to define destructor?

Options :

- a. destrucor()
- b. __destruct()
- c. __ destructor()

d. destruct()

3.The destructor method cannot accept any argument.

Options :

a. False

b. True

6.8 Inheritance

Like other object oriented programming languages feature of Inheritance is available in PHP, here one class is derived from the other class. The class which inherits the properties of another class called as child class and the class which gets inherited is called parent class.

Extends keyword is used to defined inherited class.

Sample Code 6.8.a

```
<?php
```

```
class C_Parent{
```

```
    public function Display() {
```

```
        echo "Inside Parent Class </br>";
```

```
    }
```

```
}
```

```
class C_Child extends C_Parent { // Child class inherits the properties of Parent class
```

```
    public function Show() {
```

```
        echo "Inside Child Class";
```

```
    }
```



```

}

$C_Child= new C_Child;

$C_Child->Display();

$C_Child->Show();

?>

```

Output :

```

Inside Parent Class
Inside Child Class

```

Explanation :

In the above mentioned Sample Code 6.8.a C_Child is a child class which inherits the properties of C_Parent which is parent class using keyword extends and due to this object of child class can access public member of parent class.

Consider Example of Sample Code 6.8.b where class three inherits the method of class two and class two inherits the method of class one

Sample Code 6.8.b

```

<?php

class one{

    public function Display() {

        echo "Inside one </br>";

    }

}

```

```
class two extends one { // Child class inherits the properties of Parent class
```

```
public function Show() {
```

```
    echo "Inside two </br>";
```

```
}
```

```
}
```

```
class three extends two { // Child class inherits the properties of Parent class
```

```
public function out() {
```

```
    echo "Inside three";
```

```
}
```

```
}
```

```
$obj= new three;
```

```
$obj->Display();
```

```
$obj->Show();
```

```
$obj->out();
```

```
?>
```

Output :

Inside one

Inside two

Inside three

Self Test (Multiple Choice Questions)

1. An inherited class is defined by using the _____ keyword.

Options:

- a. Extends
- b. Implements
- c. __extends
- d. _____implements

2. which properties of parent class access by child class?

Options :

- a. Private & public
- b. Private & protected
- c. Public & protected
- d. Only Public

6.9 Function Overriding

Like other object oriented programming languages in PHP also both child and parent classes have same function name as well as same number of parameters. With the help of function overriding we can change the behavior of parent class method.

Sample Code 6.9.a

```
<?php

class C_Parent{

    public function Display() {

        echo "Inside Parent Class </br>";

    }

}
```

```

    }

class C_Child extends C_Parent { // Child class inherits the properties of Parent class

    public function Display() {

        echo "Inside Child Class";

    }

}

$C_Parent=new C_Parent;

$C_Child= new C_Child;


$C_Parent->Display();

$C_Child->Display();

?>

```

Output :

Inside Parent Class

Inside Child Class

Explanation :

In above mentioned code of 6.f both child and parent class is having same method with name Display() having same number of parameter (zero parameter). And to execute the Display() method of both the classes we need to create different objects for both child and parent classes.

6.10 Access Specifiers

Like other object oriented programming languages PHP is also having access specifiers for controlling the access of member functions and properties of a class.

Following are the different access specifiers in PHP

- public
- private
- protected

public :

If we make methods or properties of a class as public then we can access these methods or properties anywhere inside the code.

public is default access specifier in PHP

Consider an example as mentioned in below code public access specifier is applied to name property

Sample Code 6.10.a

```
<?php

    class Student {

        public $name;

    }

    $obj = new Student;

    $obj->name = 'Vivek';

?>
```

Explanation :

In above code public access specifier is given to name so it can accessible in complete code,so code will not give error.

Consider an example as mentioned in below code protected and private access specifiers are applied to year and division properties respectively.

Sample Code 6.10.b

```
<?php

class Student {

    public $name;

    protected $year;

    private $division;

}

$obj = new Student;

$obj->name = 'Vivek';

$obj->year = '2019'; // ERROR

$obj->division = 'C'; // ERROR

?>
```

Output :

Fatal error: Uncaught Error: Cannot access protected property Student::\$year in /opt/lampp/htdocs/er.php:12 Stack trace: #0 {main} thrown in /opt/lampp/htdocs/er.php on line 12

Explanation :

Error generated in output for year and division properties as both the properties are having protected and private classifier respectively.

Consider an example as mentioned in below code public access specifier is applied to Display_name() method

Sample Code 6.10.c

```
<?php

class Student {

    public $name;

    public $year;

    public $division;

    function Display_name() { // bydefault it is public function

        echo $this->name="Vivek";

    }

}

$obj = new Student;

$obj->Display_name(); // OK

?>
```

Explanation :

In above code public access specifier is given to Display_name() method so it can accessible in complete code,so code will not give error.

Consider an example as mentioned in below code protected and private access specifiers are applied to Display_year() and Display_division() respectively.

Sample Code 6.10.d

```
<?php
```

```
class Student {
```

```
    public $name;
```

```
    public $year;
```

```
    public $division;
```

```
    function Display_name() { // bydefault it is public function
```

```
        echo $this->name="Vivek";
```

```
    }
```

```
    protected function Display_year() { // a protected function
```

```
        echo $this->year=2019;
```

```
    }
```

```
    private function Display_division() { // a private function
```

```
        echo $this->division="C";
```

```
    }
```

```
}
```



```
$obj = new Student;
```

```
$obj->Display_name(); // OK
```

```
$obj->Display_year(); // ERROR
```

```
$obj->Display_division(); // ERROR
```

```
?>
```

Output :

Vivek

Fatal error: Uncaught Error: Call to protected method Student::Display_year() from context " in /opt/lampp/htdocs/er.php:22 Stack trace: #0 {main} thrown in /opt/lampp/htdocs/er.php on line 22

Explanation :

Error generated in output for Display_year() and Display_division() methods as both the methods are having protected and private classifier respectively.

6.11 Interfaces

We can say that interface is higher level of abstraction. Syntax of interface is almost same as of class only one difference is that instead of class keyword interface keyword is used to define interface. Interface contains just the function prototypes no any data variables.

- While defining interface following are need to be considered :
- All the methods in the interface need to make public.
- All the methods in the interface have no implementation.
- One class can implements one or more interface

Sample Code 6.11.a

```
<?php
```

```
interface Year {                // Declaring interface
```

```

        public function FY();

        public function SY();

    }

    class Student implements Year{

        public function FY() {

            // Implementation of FY

        }

        public function SY(){

            // Implementation of SY

        }

    }

?>

```

Explanation :

Interface is declared with name Year and Student is class which implements the interface Year.

6.12 Abstract Classes

In PHP abstract class is declared using abstract keyword. It contains at one abstract method. Abstract class contains both abstract and non abstract methods.

Sample Code 6.12.a

```
<?php
```

```

abstract class ab_base // Abstract class

{

    abstract function Display(); //Abstract function


    function Show() //Non Abstract function

    {

        echo "This is non abstract function in abstract class ";

    }

}

?>

```

Sample Code 6.12.b

```

<?php

abstract class ab_base // Abstract class

{

    abstract function Display(); //Abstract function


}

class ab_Derived extends ab_base { // class extends abstract class

    function Display() {

        echo "In Derived class";

    }

}

```

```

    }

}

$obj = new ab_Derived;

$obj->Display();

?>

```

Output:

In Derived class

Explanation :

In above code abstract Display function is as abstract function which is implemented in ab_Derived class. In PHP we can not create object of an abstract class.

6.13 Static and Final Keywords

Static Keyword:

Static keyword is used to declare static methods.

Sample Code 6.13.a

```

<?php

class Student {

    public static function Display() {

        echo "Inside Static Function";

    }

}

```

```
Student::Display();
```

```
?>
```

Output:

Inside Static Function

Explanation :

Static Function Display is called by class name and double colon (::).

Final Keyword :

We can use Final keyword for both the classes and methods In PHP .

Final keyword to method:

If we declare any method with the final keyword then this particular method can not be override.

Sample Code 6.13.b

```
<?php
```

```
class C_Parent {
```

```
    final function Display() {
```

```
        echo " Base class final Display function </br>";
```

```
    }
```

```
    function Show() {
```

```

        echo "This is not final Show function in base class </br>";

    }

}

class C_child extends C_Parent {

    function Show() {

        echo "This is non final Show function in Derived class";

    }

}

$obj = new C_child;

$obj->Display();

$obj->Show();

?>

```

Output :

Base class final Display function
 This is non final Show function in Derived class

Final keyword to Class:

If we declare class as final then it can not be extend

Sample Code 6.13.c

```

<?php

final class C_Parent {          // Class declared with final keyword

```

```

final function Display() {

    echo " Base class final Display function </br>";

}

function Show() {

    echo "This is not final Show function in base class </br>";

}

}

class C_child extends C_Parent {

    function Show() {

        echo "This is non final Show function in Derived class";

    }

}

$obj = new C_child;

$obj->Display();

$obj->Show();

?>

```

Output:

Fatal error: Class C_child may not inherit from final class (C_Parent) in /opt/lampp/htdocs/er.php on line 23

6.14 Calling Parent Constructors

Constructor of parent is not called implicitly in the child class constructor. If we want to access parent class constructor in child class then we need to call `parent::__construct()`.

Sample Code 6.14.a

```
<?php

class C_Parent {

    function __construct() {

        print "In Parent Class constructor </br>";

    }

}

class C_Child extends C_Parent {

    function __construct() {

        parent::__construct();

        print "In Child Class constructor";

    }

}

$Pobj = new C_Parent;

$Cobj = new C_Child;

?>
```


Output :

In Parent Class constructor

In Parent Class constructor

In Child Class constructor

6.15 Namespaces

If we want to use same name repeatedly in same program creating Namespace is the best solution for that. We can **redeclare** the same methods or classes in the separate namespace within a single program without getting any error.

Syntax for declaring namespace is as follows

```
<?php

namespace NamespaceName {

    // We can declare Classes or Functions or interfaces

}

?>
```

6.16 Functions

Creating Function :

function keyword is required to declare function in PHP

Below mentioned is the syntax of function

```
function nameoffunction(){
    // code that need to execute inside the function
}
```

Sample Code 6.16.a

```
<?php  
  
function sample()  
  
{  
  
    echo "This is sample function";  
  
}  
  
sample(); // Calling the function  
  
?>
```

Output :

This is sample function

Function with Parameters :

Sample code 6.16.b

```
<?php  
  
function sum($a,$b) // Parameters of Function  
  
{  
  
    $c = $a+$b;  
  
    echo "Addition is $c";  
  
}  
  
sum(2,3);  
  
?>
```

Output :

Addition is 5

Function with return Value :**Sample Code 6.16.c**

```
<?php

function sum($a,$b)

{

    $c = $a+$b;

    return $c; // return value using return keyword

}

echo "Addition is ".sum(2,3);

?>
```

Output :

Addition is 5

6.17 Summary

With the help of these unit students can develop source code using object-oriented approach in PHP

6.18 Exercise:

- What is a class? How object is created in PHP?
- What are constructor and destructor in PHP?
- What is an Interface in PHP?
- What are the different data types in PHP?

- What is a final keyword in PHP and when it is used?
- Create 'stud' class in PHP. Take the RollNo, Name & MobNo of student as an input of 10 students and display it back.
- What is different types of Visibility? OR What are access modifiers?
- What is the difference between Abstract class and Interface?
- What are the advantages of object oriented programming?
- What is the relation between Classes and Objects?
- A class is defined by using the _____keyword.
Options:
 - a. function
 - b. new
 - c. class
 - d. public function

7. PHP Frameworks and Laravel

A PHP Platform is a central framework allowing us to create web applications. To put it another way, it provides structure. You'll end up saving loads of time by using a PHP Framework, avoiding the need to create repetitive code, and you'll be able to build applications easily (RAD). In this chapter, we will study the various PHP frameworks.

7.1 Introduction to Framework

A PHP Framework is a set of files, grouped together to promote development. In the standard MVC (Model-View-Controller) architecture model, the configuration of PHP Frameworks makes more sense. We distinguish your business logic (Controllers) from your front-end interface (Views) database calls (models). Model View Controller (MVC) is the general idea behind the workings of a PHP framework. MVC is a programming architectural pattern that isolates business logic from the UI, allowing one to be altered independently from the other (also known as separation of concerns). Model refers to data with MVC, View refers to the layer of presentation and Controller refers to the application or the business logic. MVC effectively breaks up an application's development process, so you can focus on individual elements while others are unaffected. In PHP, this essentially makes coding simpler and less difficult. PHP Frameworks give your PHP projects security and efficiency. We take things like SQL injections, XSS attacks, and more into account, and present fundamental protections against those risks.

Why to use PHP Framework?

For various reasons, developers should use PHP frameworks but the number one reason is to speed up the development process. Reusing code across similar projects will save a significant amount of time and effort on the developer. A system provides pre-built modules to perform repetitive coding tasks, so developers can spend their time with each and every project designing the actual application rather than re-building the base.

One big reason developers are using frameworks is consistency. Although simplicity is one of the greatest assets of PHP, and the reason that many people prefer to use this language of scripting, it may also be one of its greatest downfalls. It's relatively easy to write bad code and not even know it, particularly for beginners. The code will often always work with PHP, however unknowingly you may have created in your coding a large security hole

that could be vulnerable to attacks. It is important to remember that PHP is a very forgiving language, so making sure to tie up any loose ends in your coding is even more important – even if the code appears to work properly.

Finally, there is extensive availability of PHP frameworks, and there are lots of different frameworks to choose from. You can even build your own, though many developers want to choose from any of the most common frameworks because of their popularity, large support teams, and forums / communities that allow you to connect with other developers using the same framework. As a side note, you should always look at your project to decide first whether or not you should even use a system. There are some questions that you can ask yourself: Will it save you, and anyone else who may use it, time and effort? Will the app perform better? Will it improve stability? If you can answer yes to any of those questions, the correct answer for that particular project may be a PHP framework.

List of the best PHP frameworks,

- Laravel
- c
- CodeIgniter
- Symfony
- CakePHP
- Yii
- Zend Framework
- Phalcon
- FuelPHP
- PHPixie
- Slim

7.1.1 Laravel

Laravel is a free, open-source PHP software platform, developed by Taylor Otwell and designed to build web applications based on the architectural template model – view – controller (MVC) and based on Symfony. Some of Laravel's features are a scalable packaging framework with a dedicated dependency manager, multiple ways to access relational databases, utilities that assist in deployment and maintenance of applications, and its bias towards syntactic sugar. Laravel is an intuitive, elegant web application platform with

a syntax. We believe creation must be a fully rewarding, fun, artistic experience. Laravel tries to remove the pain from creation by relieving common tasks used in most web projects, such as authentication, routing, sessions and caching.

Laravel aims to make the development process friendly to the developer, without losing the functionality of the application. Good developers come up with the best code. To this end, we have tried to combine the very best of what we've seen in other software architectures, including frameworks implemented in other languages like Ruby on Rails, ASP.NET MVC, and Sinatra. Laravel is open but efficient, providing powerful tools that are required for stable, wide applications. A superb container reversal, articulate migration framework, and tightly integrated unit testing help give you the resources you need to develop any application you are tasked with.

7.1.2 CodeIgniter

CodeIgniter is a web platform for rapid development of open-source software for use with PHP in creating interactive websites. It is loosely based upon the development pattern of the popular model – view – controller (MVC). While controller classes within CodeIgniter are a necessary part of development, models and views are optional. CodeIgniter may also be updated to use Hierarchical Model View Controller (HMVC) which allows developers to maintain modular Controller, Model and View grouping organized in a subdirectory format.

As compared to other PHP frameworks, CodeIgniter is most often noted for its speed. In an overall critical look at PHP frameworks, PHP developer RasmusLerdorf spoke at frOSCon in August 2008, saying he liked CodeIgniter "since it is faster, lighter and the least like a frame."

7.1.3 CakePHP

CakePHP is a Web application that is open source. It implements the model – view – controller (MVC) method and is written in PHP, based on the Ruby on Rails framework, and licensed under the MIT License. CakePHP uses well-known principles in software engineering and trends in software design, such as configuration convention, model – view – controller, active record, association data mapping and front control.

CakePHP began in April 2005, when Michal Tatarynowicz, a Polish programmer, wrote a minimal version of a rapid application development framework in PHP, dubbing it Cake. He released the code under the MIT license, and opened it up to developers ' online community. As of December 2005, L. Masters G. J. Woodworth established the Cake

Software Foundation to promote CakePHP related development. As of May 2006, version 1.0 was released. One of the inspirations of the project was that of Ruby on Rails, using many of their ideas. Many sub-projects have since developed and spawned in the community. In October 2009, Woodworth project manager and developer N. Abele resigned from the project to focus on their own projects, including the Lithium Web Project (formerly part of CakePHP). The remaining development team decided to work on the aforementioned initial roadmap.

7.1.4 Yii

Yii is an open source, object-oriented Web application framework for MVC PHP component-based applications. Yii is pronounced as "Yee" or [ji:] and it means simple and evolutionary" in Chinese and can be an acronym for "Yes It Is!". Yii started as an attempt to fix perceived PRADO platform drawbacks: slow handling of complex websites, steep learning curve, and difficulty in customizing many controls. Yii is a popular platform for Web programming, meaning it can be used with PHP to build all sorts of Web applications. Its component-based architecture and sophisticated caching support make it particularly suitable for the development of large-scale applications such as portals, forums, content management systems (CMS), e-commerce projects, RESTful Web services, and so on.

Yii features include:

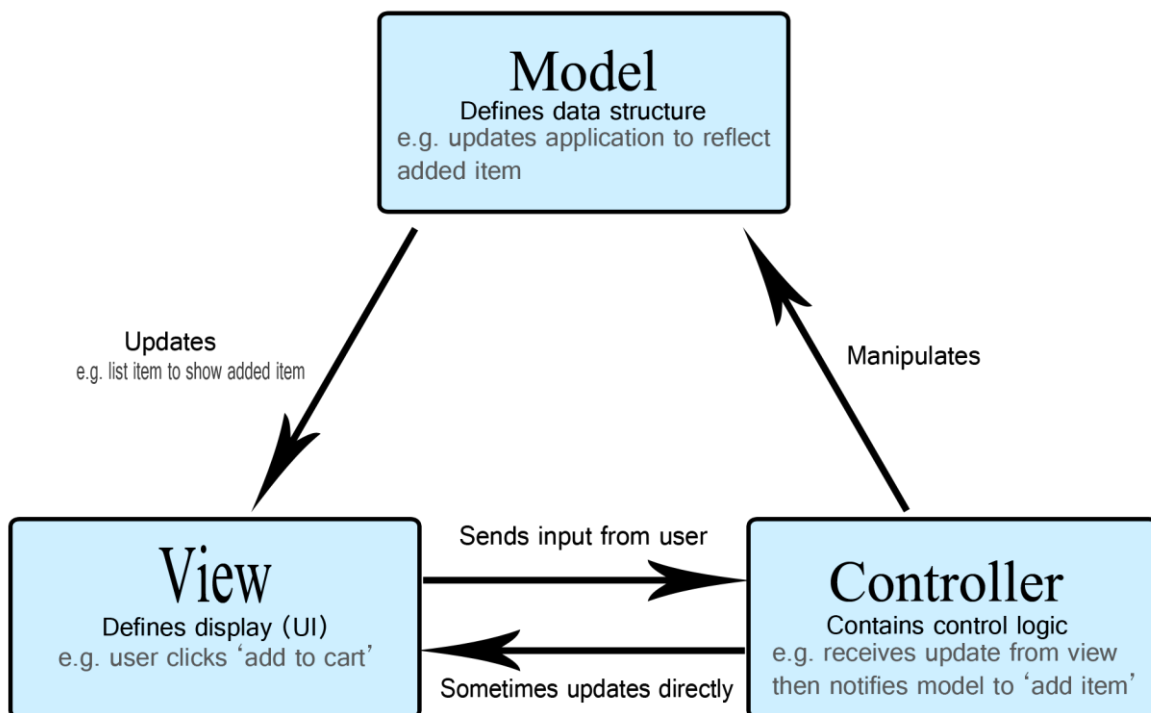
- Model-View-Controller (MVC) design pattern.
- Generation of complex WSDL service specifications and webservice request handling management.
- Internationalization and localization (I18N and L10N), including translation of texts, formatting of date and time, formatting of numbers and finding interfaces.
- Layered caching scheme supporting data caching, page caching, caching of fragments, and dynamic content. The data caching medium can be changed.
- Handling and logging errors You can categorize, filter, and route the log messages to different destinations.
- Security measures include avoidance of cross-site scripting (XSS), cross-site request forgery (CSRF) and interfering with cookies.
- Testing of unit and functions based on PHPUnit and Selenium.
- Automatic generation of code for skeleton application, CRUD apps, via the Gii tool.
- Code generated by Yii components and command line tools is XHTML compliant.

- Designed to function well with third party technology. For example, code from PEAR or the Zend Framework may be included.

7.1.5 MVC (Model View controller)

7.1.5.1 Introduction

Model – View – Controller (usually known as MVC) is a pattern of software design commonly used to develop user interfaces that divide the associated program logic into 3 interconnected elements. This is done to distinguish internal information representations from the way information is presented to and approved by the user. Traditionally used for graphical user interfaces (GUIs) on desktops, this pattern has become popular for web app design. Popular programming languages such as JavaScript, Python, Ruby, PHP, Java, and C# have MVC frameworks which are used in developing web applications.



Model – It is the pattern's central component. It is the dynamic data structure of the program, independent of user interface. It manages the application's data, logic, and rules directly.

View – It is any sort of information representation, such as a map, diagram, or table. Multiple views of the same information, such as a management bar chart and a tabular view for accountants are possible.

Controller – It takes data, and transforms it to model or display commands.

The model – view – controller architecture, in addition to dividing the application into those components, determines the interactions between them.

- The model is responsible for handling application data. It receives controller user input.
- The view implies the model is viewed in a particular format.
- The controller reacts to user input and performs interactions on objects of the data model. The controller receives the data, validates it internally and then transfers the input onto the computer.

7.1.5.2 Goals of MVC

Simultaneous Development

Because MVC decouples the different components of an application, developers can work on different components in parallel, without affecting or blocking each other. For example, a team could be dividing their developers between the front end and the back end. Back-end developers can build the data structure and how the user interacts with it, without having to complete the user interface. Conversely, the front-end developers will design and check the application's interface before the data structure is available.

Code Reuse

It is possible to refactor the same (or similar) view for one application with different data for another application, because the view is simply handling how the data is displayed to the user. Unfortunately this does not work when this code is also useful for user input handling.

7.1.5.3 Advantages & Disadvantages

Advantages

- Simultaneous development – Multiple developers can work on the model, controller and views at the same time.
- High cohesion – MVC allows for the logical grouping of similar actions together on a controller. There are also groupings of views for a particular model.
- Loose coupling – The very nature of the MVC framework is such that models, views or controllers are loosely coupling.
- Ease of modification – Future development or modification is simpler, because of the separation of responsibilities.
- Multiple views for a model – Models can have many different views.

Disadvantages

The drawbacks of MVC for wrongly factored applications may usually be defined as overhead.

- Code navigability – The navigation system can be challenging because it introduces new levels of abstraction and requires users to conform to MVC's requirements for decomposition.
- Multi-artifact consistency – The decomposition of a feature into three artefacts causes dispersion. Thus, require developers to keep multiple representations consistent at once.
- Undermined by inevitable clustering – Applications appear to communicate strongly between what the user sees and what the user is using. Hence, the computation and state of each function appears to be clustered into one of the 3 program parts erasing MVC's purported advantages.
- Excessive boilerplate – Because the application computation and state are typically clustered into one of the 3 parts, the other parts degenerate into either boilerplate shims or code-behind which only exist to satisfy the MVC pattern.
- Pronounced learning curve – Multi-technology knowledge becomes norm. Developers who use MVC need to be experienced in various technologies.
- Lack of incremental benefit – UI implementations are already factored into modules, gaining code reuse and independence through the design of components, leaving MVC no incremental gain.

7.2 Laravel Installation

7.2.1 Server Requirements

There are a few program specifications to the Laravel framework. The Laravel Homestead virtual machine meets all of these criteria, so it is highly recommended that you use Homestead as your local development environment in Laravel.

If you don't use Homestead, however, you will need to ensure that your server meets the following requirements:

- PHP \geq 7.1.3
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- BCMath PHP Extension

7.2.2 Installing Laravel

Laravel manages its dependencies using Composer. So, make sure you have Composer installed on your machine, before using Laravel.

Using Laravel Installer

First, use Composer to download Laravel installer:

```
composer globalrequirelaravel/installer
```

Make sure you place the system-wide vendor bin directory for the composer in your \$PATH so that your system can locate the Laravel executable. Based on your operating system, this directory exists at various locations; however, some common locations include:

- macOS: \$HOME/.composer/vendor/bin
- GNU / Linux Distributions: \$HOME/.config/composer/vendor/bin

When mounted, the `laravel new` command will establish a fresh installation of Laravel in the directory you choose. For example, `laravel new web` will create a `web` named directory containing a fresh Laravel installation already installed with all of the Laravel dependencies:

```
laravel new web
```

Using Composer Create-Project

Alternatively, you can also install Laravel via the `create-project` Composer command in your terminal:

```
composer create-project --prefer-distlaravel/laravel web "5.7.*"
```

Local Development Server

If you have PHP installed locally and want to use the built-in development server for your application, you can also use the `serve` Artisan command. This command will open a server for development at `http://localhost:8000`:

```
php artisan serve
```

Configuration

Public Directory

You should configure the document / web root of your web server after installing Laravel to be the `public` directory. In this directory the `index.php` acts as the front controller for all HTTP requests that access your domain.

Configuration Files

All Laravel application configuration files are mostly stored in the `config` directory. Each option is documented so be free to browse through the files and familiarize yourself with the options available.

Directory Permissions

You may need to configure certain permissions once you have installed Laravel. Storage directories and `bootstrap/cache` directories should be writable through your web server, or Laravel won't run. If you are using the virtual Homestead system, you should already set these permissions.

Application Key

After installing Laravel, the next thing you should do is set your application key to a random string. If you have installed Laravel via Composer or the Laravel installer, the `php artisan key:generate` command has already set this key for you.

This string will normally have to be 32 characters long. The key can be set within the file for the `.env` environment. If the `.env.example` file has not been renamed to `.env`, you should do that now. **If the application key is not set it will not be secure for your user sessions and other encrypted data!**

Additional Configuration

Almost no other configuration from the box is required by Laravel. You're free to begin development! You may want to test the `config / app.php` file and its documentation though. This includes many choices that you may want to modify according to your query, such as `timezone` and `locale`.

Web Server Configuration

Pretty URLs

Apache

Laravel provides a `public/.htaccess` file that is used without the path front controller `index.php` to provide URLs. Until serving Laravel with Apache, make sure the `mod_rewrite` module is allowed so that the server honors the `.htaccess` script.

If your Apache installation does not work with the `.htaccess` file that is shipping with Laravel, try this alternative:

```
Options +FollowSymLinks-Indexes
RewriteEngine On

RewriteCond %{HTTP: Authorization}.
RewriteRule .*-[E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]

RewriteCond %{REQUEST_FILENAME}!-d
RewriteCond %{REQUEST_FILENAME}!-f
RewriteRule ^index.php [L]
```

Nginx

If you use Nginx, all requests will be directed to the `index.php` front controller via the following directive in your site configuration:

```
location /{
    try_files$uri$uri//index.php?$query_string;
}
```

7.3 Laravel Database Connectivity

- Configuration
- Read / Write Connections
- Running Queries
- Database Transactions
- Accessing Connections
- Query Logging

Configuration

Laravel makes it extremely simple to connect to databases, and to run queries. The database configuration file is `app/config/database.php`. You can define all of your database connections in this file, and specify which connection should be used by default. The file provides examples for all of the available database systems.

Currently Laravel supports four database systems: MySQL, Postgres, SQLite, and SQL Server.

Read / Write Connections

Sometimes you might want to connect one database to `SELECT` statements, and another for `INSERT`, `UPDATE`, and `DELETE` statements. Laravel makes this a breeze, and it will always use the proper connections whether you're using raw queries, the query builder or the Eloquent ORM. To see how to configure read / write connections, let's take a look at this example:

```
'mysql'=>array(
    'read'=>array(
        'host'=>'192.168.1.1',
    ),
    'write'=>array(
```

```

        'host'=>'196.168.1.2'
    ),
    'driver'=>'mysql',
    'database'=>'database',
    'username'=>'root',
    'password'=>'',
    'charset'=>'utf8',
    'collation'=>'utf8_unicode_ci',
    'prefix'=>'',
),

```

Note that the configuration list has been introduced with two keys: `read` and `write`. Both of these keys have a single key sequence values: `host`. The rest of the read and write connection database options will be merged from the main `mysql` array. So, if we want to override the values in the main array, we need only place items in the `read` and `write` arrays. So, in this case, `192.168.1.1` will be used as the "read" connection, while `192.168.1.2` will be used as the "write" connection. The credentials of the database, prefix, character set, and all other options in the main `mysql` array will be shared over both connections.

Running Queries

Once you have configured the link to your database, you may use the `DB` class to run queries.

Running A Select Query

```

$results=DB::select('select * from users where id =
                    ?',array(1));

```

The `select` method will always return an array of results.

Running An Insert Statement

```

DB::insert('insert into users (id, name) values (?,
          ?)',array(1,'Parag'));

```


Running An Update Statement

```
DB::update('update users set count = 100 where name =  
            ?',array('Sneha'));
```

Running A Delete Statement

```
DB::delete('delete from users');
```

Note: The `update` and `delete` statements will return the number of rows affected by the operation.

Running A General Statement

```
DB::statement('drop table users');
```

Listening For Query Events

You may listen for query events using the `DB::listen` method:

```
DB::listen(function($sql,$bindings,$time)  
{  
    //  
});
```

Database Transactions

You may use the `transaction` method to execute a series of operations within a database transaction:

```
DB::transaction(function()  
{  
    DB::table('users')->update(array('count'=>1));  
  
    DB::table('posts')->delete();  
});
```

Note: Any exception thrown into the closing of the transaction would cause the transaction to be immediately rolled back.

You may need to start a transaction sometimes by yourself:

```
DB::beginTransaction();
```

A transaction can be rolled back using `rollback` method:

```
DB::rollback();
```

Finally, the `commit` method allows you to commit a transaction:

```
DB::commit();
```

Accessing Connections

You can access these via `DB::connection` method when using multiple connections:

```
$users=DB::connection('foo')->select(...);
```

You can also access the raw PDO case, which underlies:

```
$pdo=DB::connection()->getPdo();
```

You might need to reconnect to a given database sometimes:

```
DB::reconnect('foo');
```

If you want to disconnect from the specified database due to more than the underlying PDO instance's `max_connections` limit, use the `disconnect` method:

```
DB::disconnect('foo');
```

Query Logging

By default Laravel keeps a log of all queries running for the current request in memory. However, in some cases this can cause the application to use excess memory, such as when inserting a large number of rows. To disable the log, you may use the `disableQueryLog` method:

```
DB::connection()->disableQueryLog();
```

You can use `getQueryLog` method to get an array of the queries you have executed:

```
$queries=DB::getQueryLog();
```

Exercise:

1. What is framework? List various PHP frameworks.
2. Explain the reason to use the PHP frameworks.
3. Explain MVC along with its goal.
4. State the various advantages and disadvantages of MVC.
5. List the system requirement for the installation of the Laravel framework.
6. State the various steps involved in the installation of the Laravel framework.

8. Introduction to CMS

In this chapter, we are going to study the concept of Content Management System (CMS), various CMS tools used to design the website, installation of WordPress, website creation using WordPress, etc.

8.1 Introduction

A content management system or CMS is a program that makes content easier to create, edit, organize, and publish. WordPress is a content management system that enables you to build your content and publish it on the internet. While being mostly used for web publishing, it can be used to handle content on an intranet, or on a single computer.

WordPress allows users to have complete control over the files, documents, and interface design and display. To publish content using WordPress, you do not need to know a single line of code. The beauty of a good content management system is to allow any user with no technical know-how to build and manage their contents.

Average user or small business in the earlier days had to rely on static HTML pages because they couldn't afford a content management system that would cost hundreds of thousands of dollars. That problem has now been solved. WordPress is free and open source for anybody to use.

WordPress is used creatively in all manner of ways. We've seen WordPress being used to power small business websites, forums, large university websites, portfolios, real estate listing platform, company internal communication network, online archives, film repositories, application technology base, arcade pages, and anything else you may think of.

There are various content management systems that will provide the features mentioned above. Some of them,

- WordPress
- Joomla
- Drupal
- Magento

8.1.1 WordPress

WordPress is a free and open-source content management system (CMS) based on PHP & MySQL. The features include a design module and a prototype framework. It is mostly related to blogging but embraces other forms of web content including more conventional mailing lists and forums, media galleries, and online stores. Used as of April 2019 by over 60 million websites, including 33.6 percent of the top 10 million websites, WordPress is the most popular website management system in use. WordPress was also used in other application areas, such as omnipresent display systems (PDS).

WordPress was launched May 27, 2003 as a b2/cafeblog fork by its owners, Matt Mullenweg and Mike Little. The software is published (or later) under the GPLv2 license. To function, WordPress has to be installed on a web server, either as part of an Internet hosting service such as WordPress.com or as part of a computer running the WordPress.org software package to act as a network host in its own right. A local machine may be used for the research and learning purposes of a single user.

8.1.2 Joomla

Joomla is a free and open source content management system (CMS) created by Open Source Matters, Inc. for the publication of web content. It is built on a web application framework for model – view – controller which can be used independently of the CMS.

Joomla writes in PHP, using object-oriented programming techniques (since version 1.5) and software design patterns, stores data in a database called MySQL, MS SQL (since

version 2.5), or PostgreSQL (since version 3.0), and includes features such as web loading, RSS feeds, printable page versions, news alerts, forums, search and language internationalization support.

There are more than 8,000 free and commercial extensions available from the official Joomla Extensions Directory, and more from other sources. It is estimated to be the second most frequently used online content management system, after WordPress.

8.1.3 Drupal

Drupal is a free and open-source content management system developed under the GNU General Public License and distributed under PHP. Drupal provides at least 2.3 per cent of all websites worldwide with a back-end framework – ranging from personal blogs to corporate, political, and government sites. Drupal is also used by systems for knowledge management and for business collaboration.

The Drupal ecosystem comprised more than 1,37 million members as of March 2019, including 114,000 actively contributing users, resulting in over 42,650 free modules which extend and customize the Drupal functionality, more than 2,750 free themes that modify the look and feel of Drupal, and at least 1,270 free distributions that allow users to set up a complex, usage-specific Drupal quickly and easily in fewer steps.

Drupal's initial version, known as the Drupal core, includes basic features common to content-management systems. These include registration and maintenance of user account, menu management, RSS feeds, taxonomy, configuration of the page layout, and system management. The installation of the Drupal core can be a basic website, a single- or multi-user blog, an internet forum, or a community website offering user-generated content.

Drupal also describes itself as a platform for Web applications. As contrasted with popular frameworks, Drupal meets most of the functionality specifications generally accepted for such web frameworks.

Drupal runs on any computing platform that supports a web server that can run PHP, as well as a database that can store content and configuration.

8.1.4 Magento

Magento is an e-commerce platform open-source, written in PHP. The software was originally developed with the assistance of volunteers by Varien, Inc, a US private company headquartered in Culver City, California.

On 31 March 2008, Varien released the software's first general-availability update. Roy Rubin, Varien's former CEO, later sold a large share of the company to eBay, which ultimately purchased it completely and then sold it to Permira; Permira subsequently sold it to Adobe.

Magento 2.0 was released on 17th November 2015. Among the features which have changed in V2 are: Reduced table locking problems, improved page caching, enterprise-grade scalability, rich built-in data snippets, new file structure with easier customization, CSS Preprocessing using a resolution of the LESS & CSS URL, improved performance and a more organized code base. Magento uses the relational database management system MySQL or MariaDB, the programming language for PHP, and the Zend Framework elements. It applies object-oriented programming conventions and model – view – controller architecture. In addition, Magento uses the model entity – attribute – value to store data. In addition, using the JavaScript library Knockout.js, Magento 2 introduced the Model-View-View model pattern to its front-end code.

8.2 WordPress

8.2.1 History and Overview

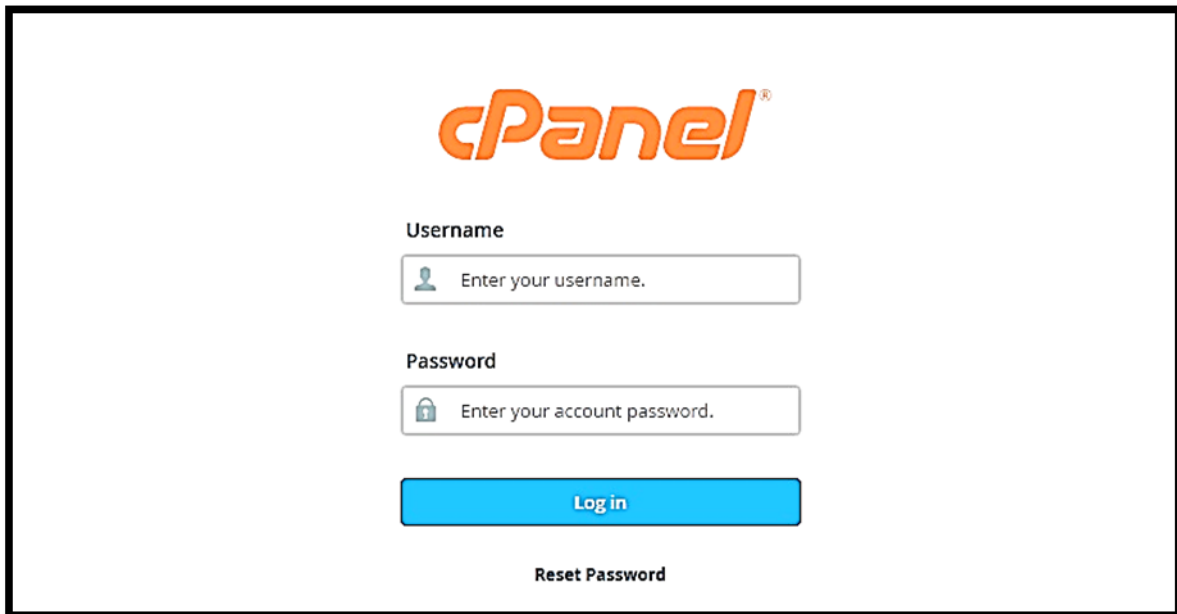
A predecessor of WordPress was b2/cafelog, more commonly known as b2 or cafelog. As of May 2003, b2/cafelog was reported to have been built on around 2000 blogs. It was written in PHP by Michel Valdrighi, who is now a contributing WordPress developer, for use with MySQL. Though WordPress is the official successor, there is also another project, b2evolution, in active development. In 2003, WordPress first appeared as a joint effort between Matt Mullenweg and Mike Little to build a b2 fork. The word WordPress was coined by Christine Selleck Tremoulet, a colleague of Mullenweg. As of June 2019, 60.8 percent of all websites whose content management system is established are using WordPress. This represents 27.5 percent of the top ten million websites.

"WordPress is a webpage-making factory" is a core analogy intended to clarify what WordPress is and does. It stores your content that allows you to create and publish web pages that only require a domain and a working hosting site. WordPress uses a template processor to have a Web design program. The architecture is a front controller, which routes all non-static URI requests into a single PHP file that parses the URI and defines the target page. This enables more human-readable permalinks to be supported.

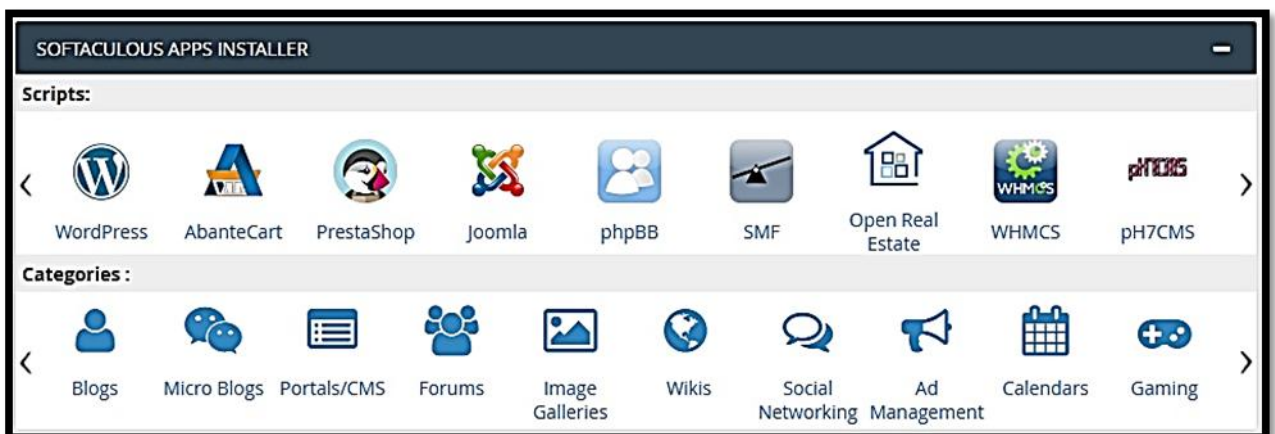
8.2.2 Installation

WordPress is very easy and simple to install. Through the hosting, we'll see step by step process of installing on the live domain.


Step 1: Login to the cPanel of the website where you want to install the WordPress.



Step 2: After successful login, cPanel dashboard will get open. At the bottom of this dashboard, there is tab as Softaculous Apps Installer. Click on the WordPress under this tab.

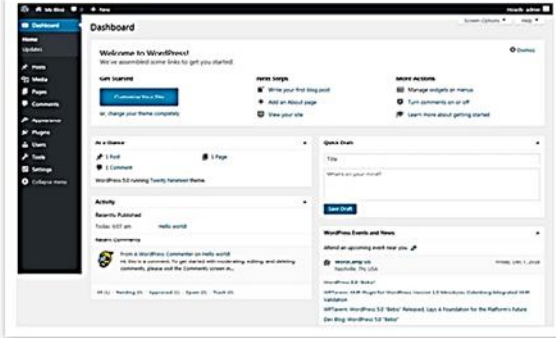


Step 3: Click in *Install* button.


WordPress

★★★★★
Version : 5.2.2, 5.1.1, 5.0.4, 4.9.10
Release Date : 18-06-2019

Install
Overview
Features
Screenshots
Demo
Ratings
Reviews
Import



WordPress is web software you can use to create a beautiful website or blog. We like to say that WordPress is both free and priceless at the same time.

The core software is built by hundreds of community volunteers, and when you're ready for more there are thousands of plugins and themes available to transform your site into almost anything you can imagine.


Over 60 million people have chosen WordPress to power the place on the web they call "home" we'd love you to join the family.

WordPress is an [Open Source](#) project, which means there are hundreds of people all over the world working on it. (More than most commercial platforms.) It also means you are free to use it for anything from your recipe site to a Fortune 500 web site without paying anyone a license fee.

Install Now
My Apps
Go to Settings to activate Windows.
Software Support

Space Required

Step 4: Fill all the relevant information on the setup page. Remember to keep *Directory* empty under Software Setup part. Also select the theme then click on *Install* button.


WordPress

★★★★★
Version : 5.2.2, 5.1.1, 5.0.4, 4.9.10
Release Date : 18-06-2019

Install
Overview
Features
Screenshots
Demo
Ratings
Reviews
Import

Software Setup

Quick Install

Choose the version you want to install
Please select the version to install.
5.2.2

Choose Installation URL
Please choose the URL to install the software
https://
Choose Protocol
Choose Domain
In Directory

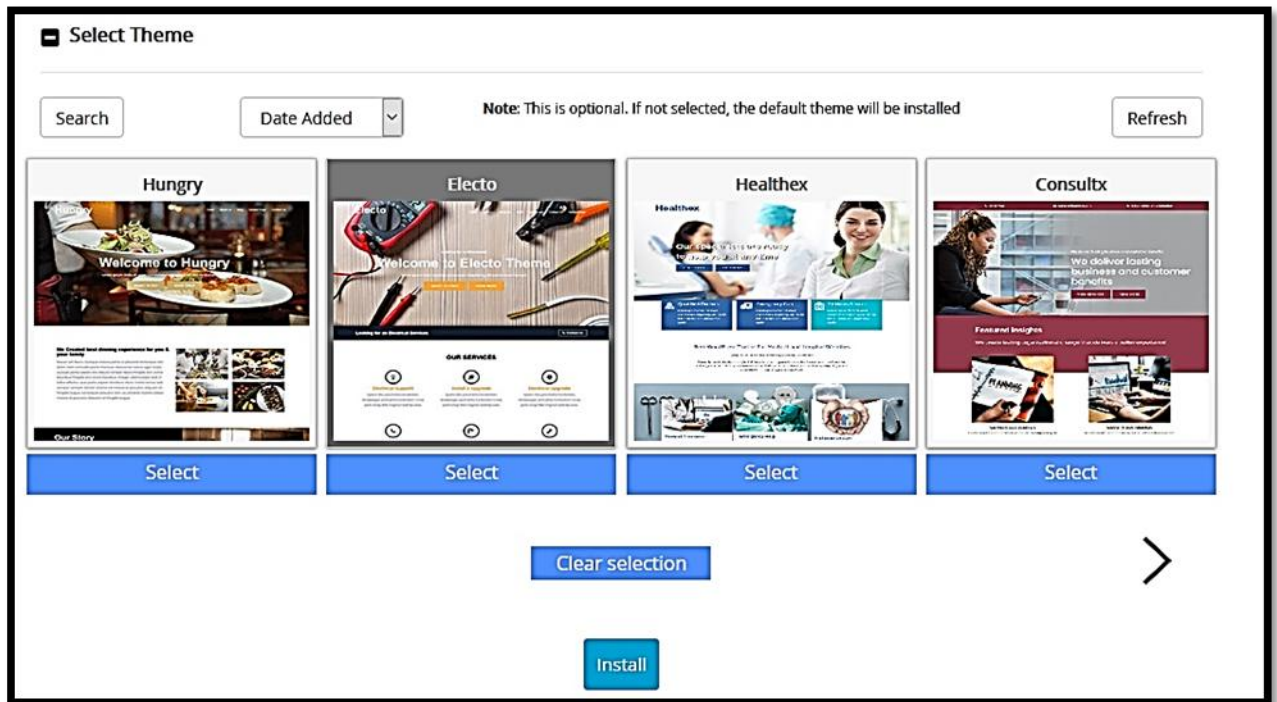
Site Settings

Site Name
My First Website

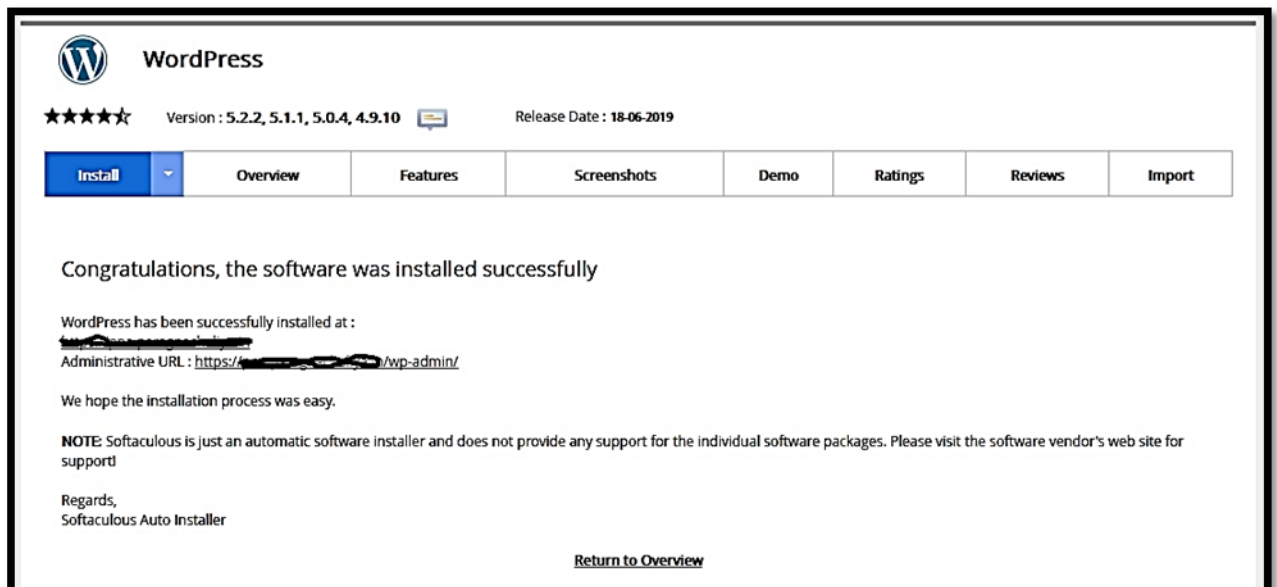
Site Description
WordPress

Enable Multisite (WPMU)
☐

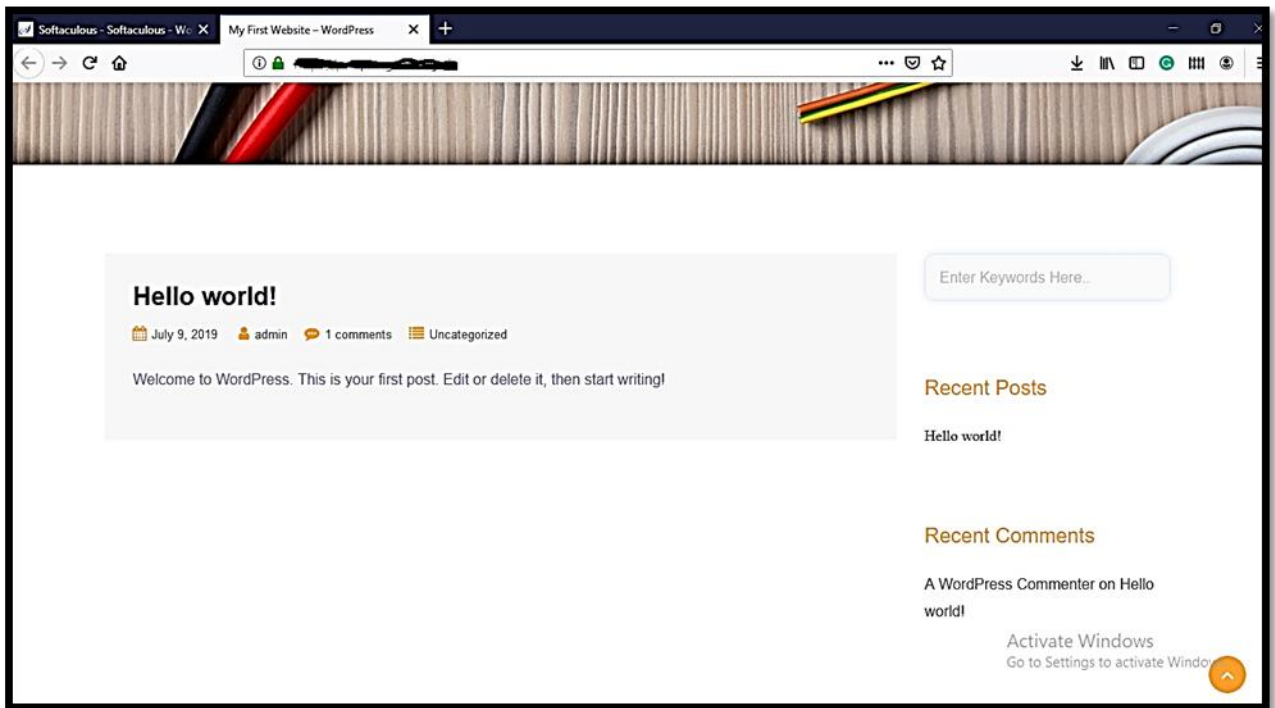
Activate Windows
Go to Settings to activate Windows.



Step 5: WordPress will be installed successfully after the completion of above step & the following window will appear showing the same message.



Step 6: Now you can see that the WordPress website is ready to use & edit.



8.2.3 Dashboard

The Dashboard's key idea is to give you a place to get an at-a-glance rundown of what's happening to your site. You can catch up on news, view your draft articles, see who's linking to you or how popular your content has been, easily message a no-frills post, or check your latest comments out and moderate. It's like an eye view of operations from a bird, from which you can swoop down into the specific details. The Dashboard contains the following modules:

- At a Glance
- Quick Draft
- Activity
- Your Stuff
- What's Hot
- Stats

At a Glance

The module At a Glance is just as it sounds! It gives a "at-a-glance" look at the posts, sites, reviews, theme and spam posts on your blog. Click on the links, and you will be taken to the corresponding screen. Akismet's caught track of your total comments and spam. You can also click on the numbers to load the screen with relevant comments.



Quick Draft

Quick Draft is a mini-post editor, which allows the Dashboard to create instant content. In the post you can include a title and body text, and save it as a Draft. You should use the Add New Post screen for additional options, such as adding categories or setting a future publication date. Below you will see links to your most recent drafts, allowing a one-click Dashboard access. When you click on any of them, editing the post should take you straight away.

Quick Draft

Title

What's on your mind?

Save Draft

Activity

The module has many new features that make working with the Dashboard's comments quick and easy.

Activity	
Recently Published	
Mar 31st, 10:39 am	Post #1
Mar 26th, 5:18 pm	Post #2
Mar 19th, 3:49 pm	Post #3
Mar 14th, 1:36 pm	Post #4
Mar 13th, 10:29 am	Post #5

Your Stuff

On WordPress.com, the **Your Stuff** module shows links to your recent activity. The module will display links to comments you have left on other WordPress.com blogs, as well as links to posts on any of your blogs where you have made changes recently.

What's Hot

What's Hot is a module that shows links to:

- Top WordPress.com blogs

- Latest Posts
- Top Posts from around WordPress.com
- Recent posts from the WordPress.com News blog

Stats

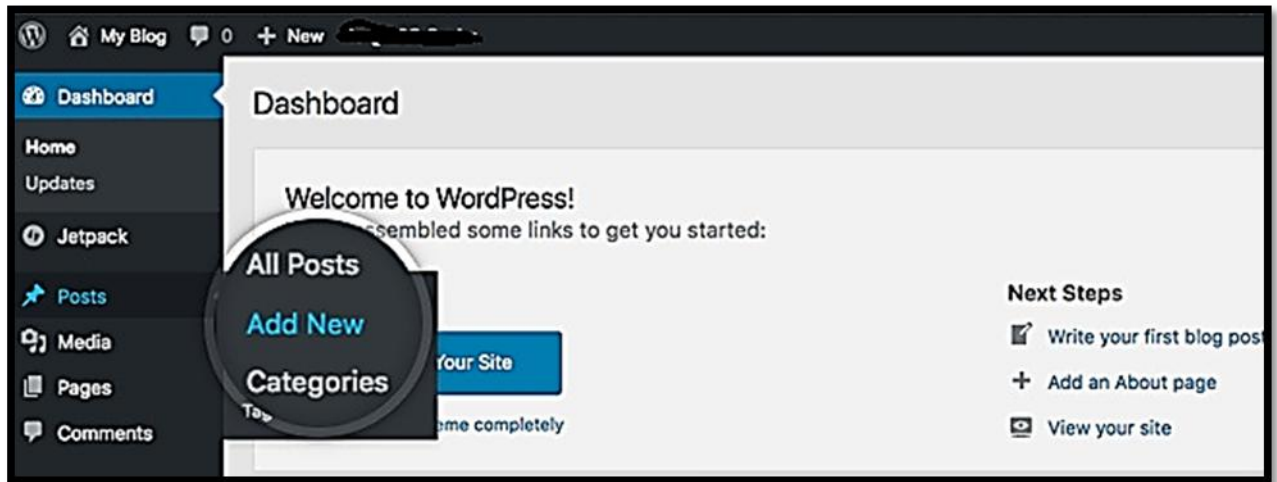
The module **Stats** is a favorite among many users. It will show you a graph of the traffic in your blog and links to some popular areas of your site. The graph simply works like the graphs on the Site Stats screens, so you can click a point to see more information about the traffic of that day.

8.2.4 Add and Publish Post

WordPress Posts are entries that appear in reverse chronological order on the home page of the blog or on the posts page. If any sticky posts are made, they will appear before the other posts. You can find posts in the Archives Categories Recent Post, and other widgets. Posts are shown in the blog's RSS feed, too. In Reading Settings, you can control how many posts are shown at one time.

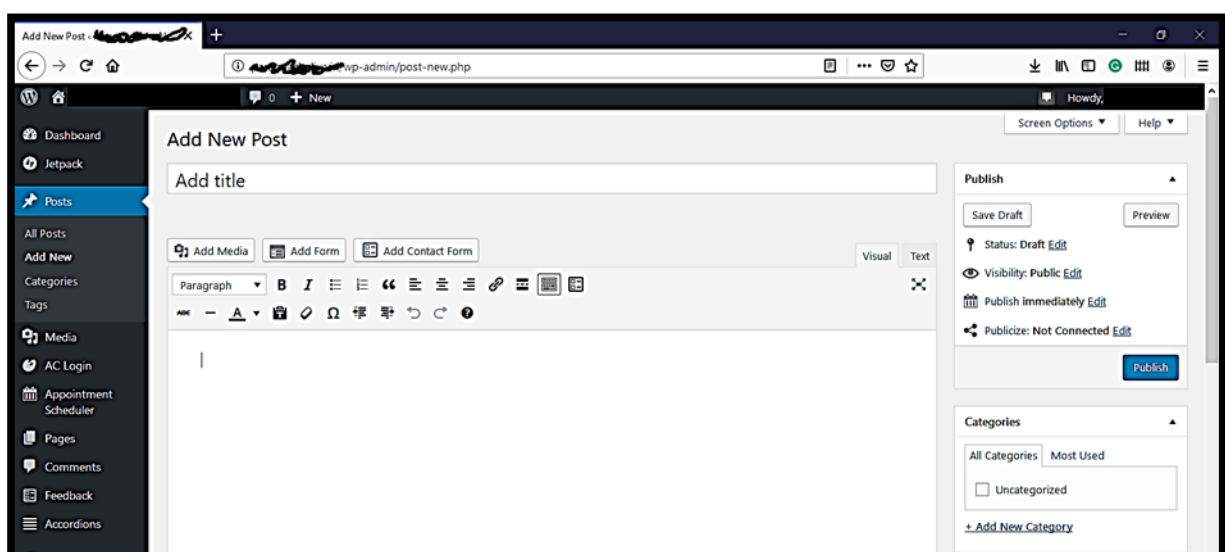
Add WordPress Post

Step 1: You must first log in to your site's wp-admin panel, then go to **Posts -> Add New**.

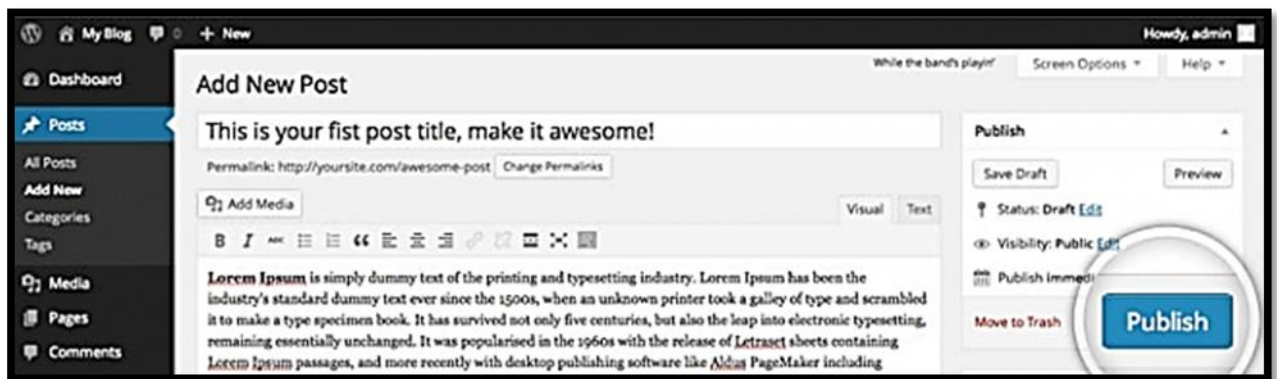


Step 2: You'll see the WordPress posts editor on this page. The core parts of this page are:

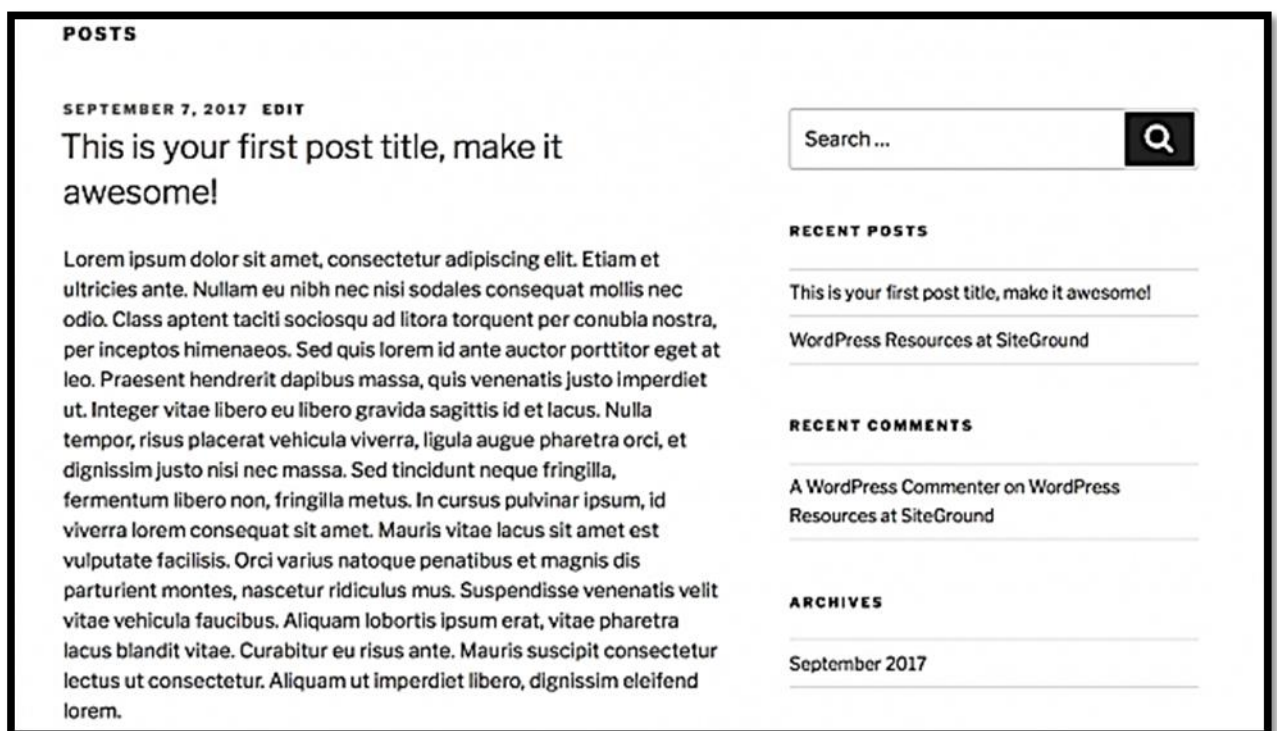
- **Post Title** - In this field enter the title of your post. It will show above your content on your theme.
- **Post Content** - You can add the actual content of your post to WordPress WYSIWYG editor. Note that it does have two tabs - **Visual** (To format your text, use the editor) and **Text** (Directly add your HTML code).



Step 3: You need to publish it once you enter your first WordPress post's content. Publishing is getting your post onto your website.



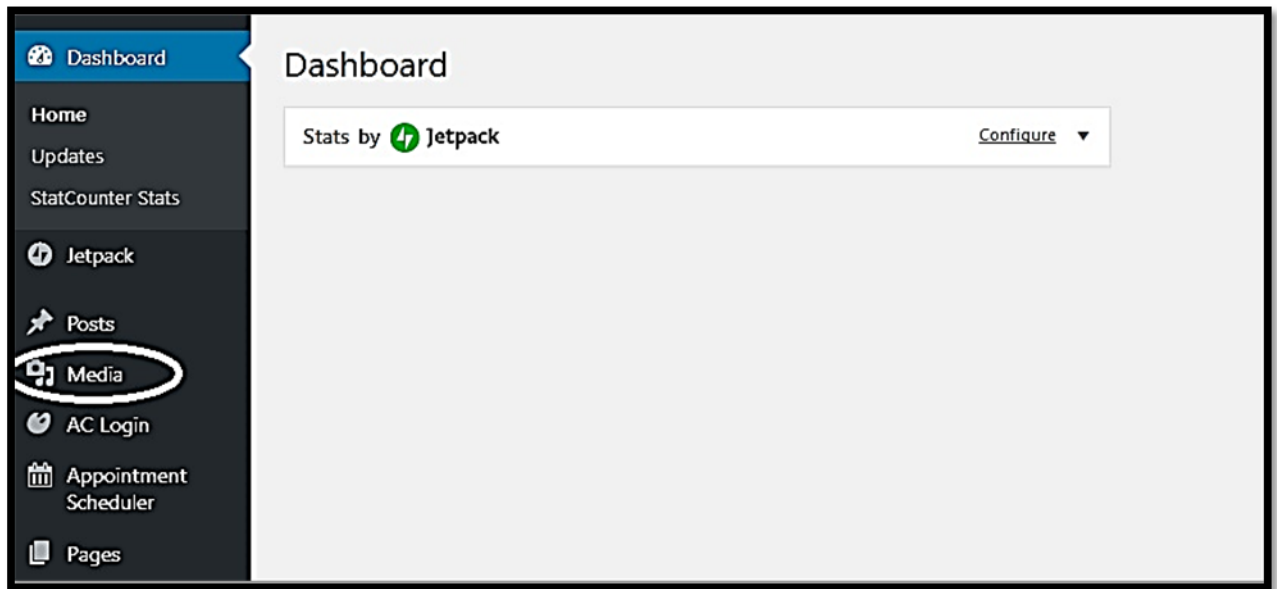
Step 4: You can go to your site's front page now to check out the newly created blog post.



8.2.5 Media Library

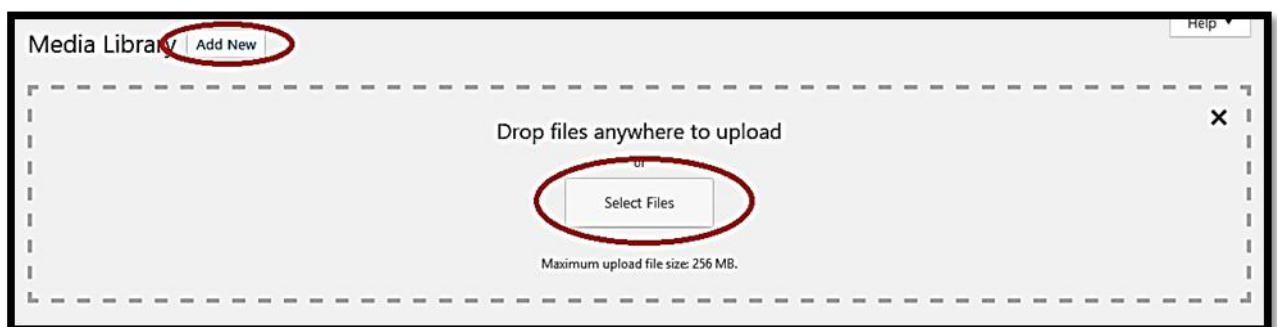
The **Media Library** is where all of your images audio, videos and documents can be managed in one location.

Navigate to the Media Library



Add Files from Computer

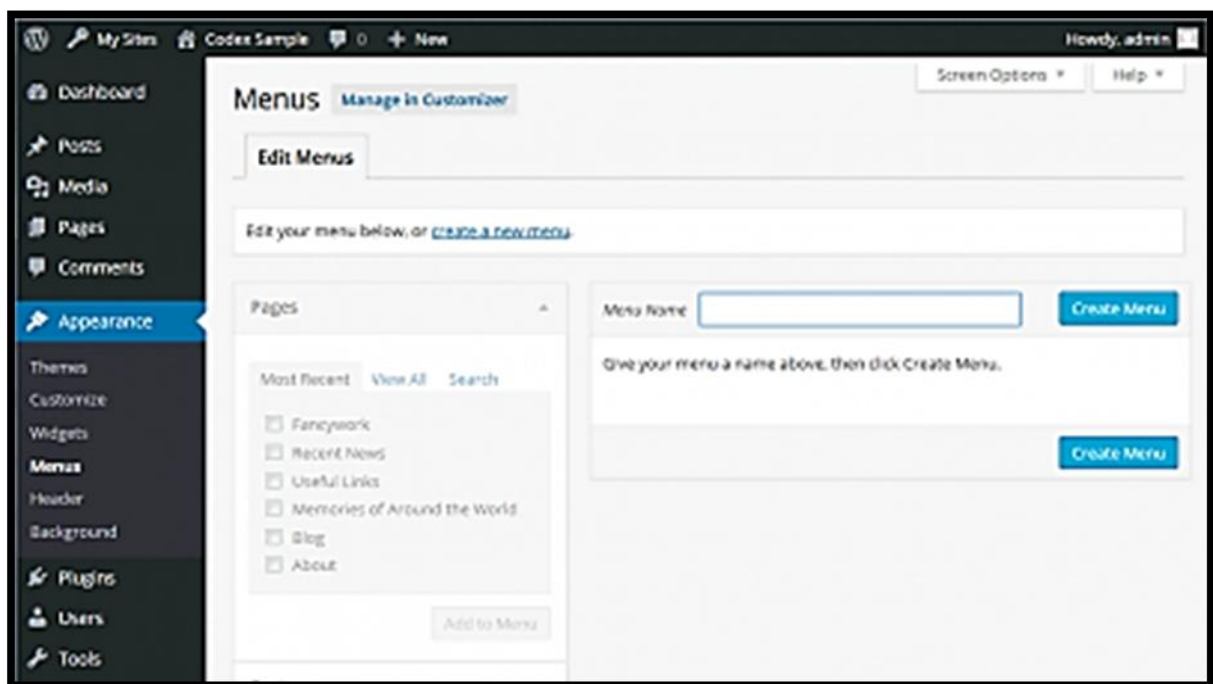
Click **Add New** to select the files you want to upload, or drag and drop files directly onto the page to add files from your computer.



8.2.6 Creating a Menu

Menu must be defined before various items are added to it. Menu can be created in WordPress using given steps.

1. Sign in to Dashboard of WordPress.
2. Select the **Menus** option to bring up the Menu Editor from the **Appearance** menu on the left-hand side of the Dashboard.
3. Choose **New Menu** at the top of the page
4. In the Menu Name box type a name for your new menu
5. Click the **Create Menu** button.



Adding Items to Menu

Different types of links can be added to the menu; these are divided between panes left of the currently edited menu.

1. Locate the **Pages** Pane.

2. Select the **View All** link to create a list of all the currently published Pages on your site within this window.
3. Tap the **checkbox** next to the title of each page to pick the Pages you want to add.
4. To add your selection(s) to the menu you created in the previous step, click the **Add to Menu** button located at the bottom of this window.
5. Once you have added all the menu items you want, press the **Save Menu** button.

Deleting Menu Item

1. Locate the menu item you wish to delete in the menu editor window
2. To expand it, press the arrow icon in the top right corner of the menu item / window.
3. Click the Link to Remove. The menu item / box shall be deleted immediately.
4. To save your changes, press the **Save Menu** button.

Creating Multi-Level Menus

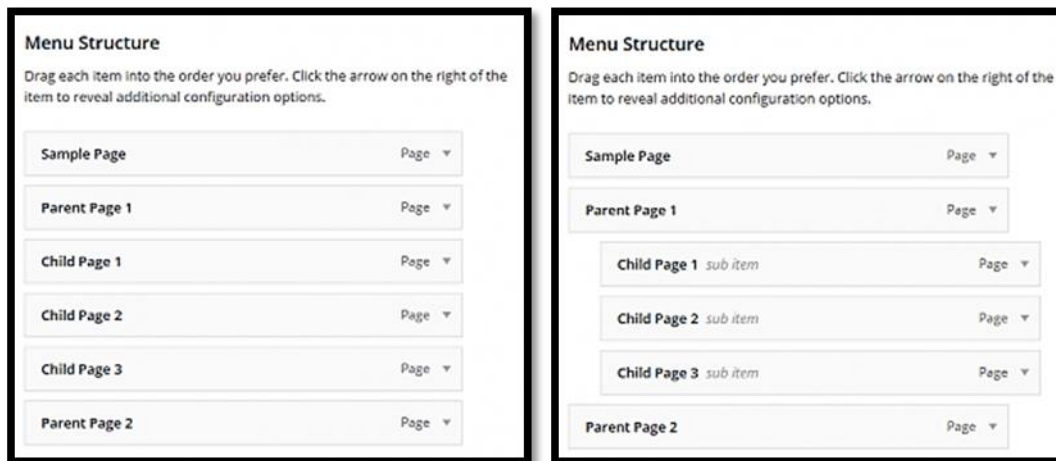
When planning menu structure it helps to consider each menu item as a heading in a formal report document. In a formal report, the headings of the main section (Level 1 headings) are the closest to the left of the page; the headings of the sub-section (Level 2 headings) are slightly further to the right; any other subordinate headings within the same section (Level 3, 4, etc.) are further to the right.

The WordPress menu editor allows a quick 'drag and drop' interface to construct multi-level menus. Drag up or down the menu items to change their appearance in the menu. To create sub-levels within your menu, drag the menu items left or right.

You need to place the 'child' underneath its 'parent' and then move it slightly to the right to make one menu item a subordinate of another.

1. Place the mouse over the menu item 'Child'
2. Drag it to the right while holding the left mouse button.

3. Release mouse button.
4. For each sub-menu item, repeat these steps
5. To save your changes, press the Save Menu button in menu editor.



After completing above mentioned steps, your newly created website will get ready on the specified domain or locally. You can see the website by entering the URL in address bar of your browser.

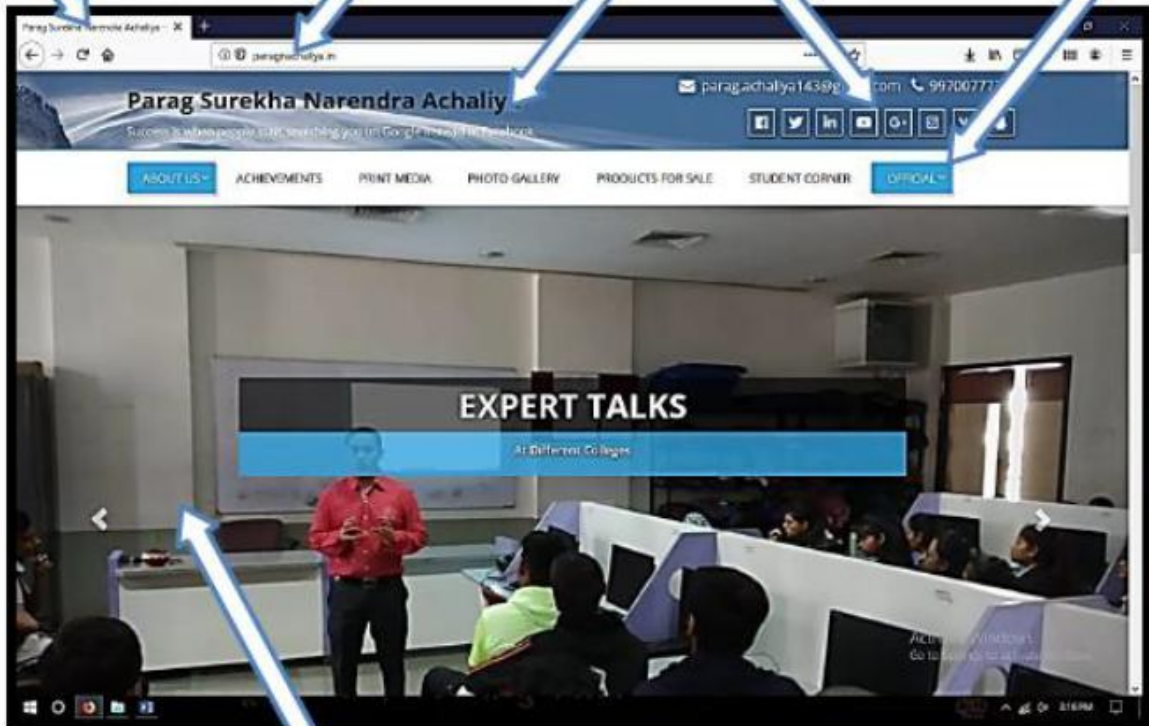
your browser.

Address Bar

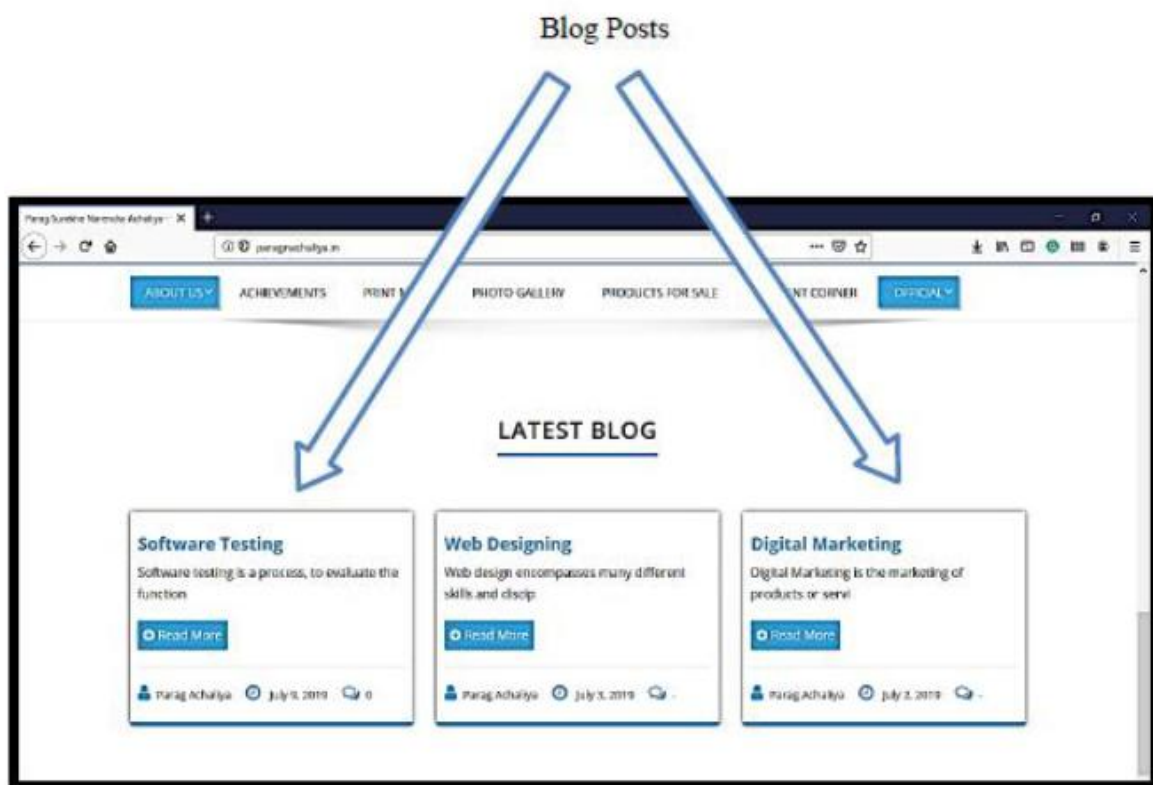
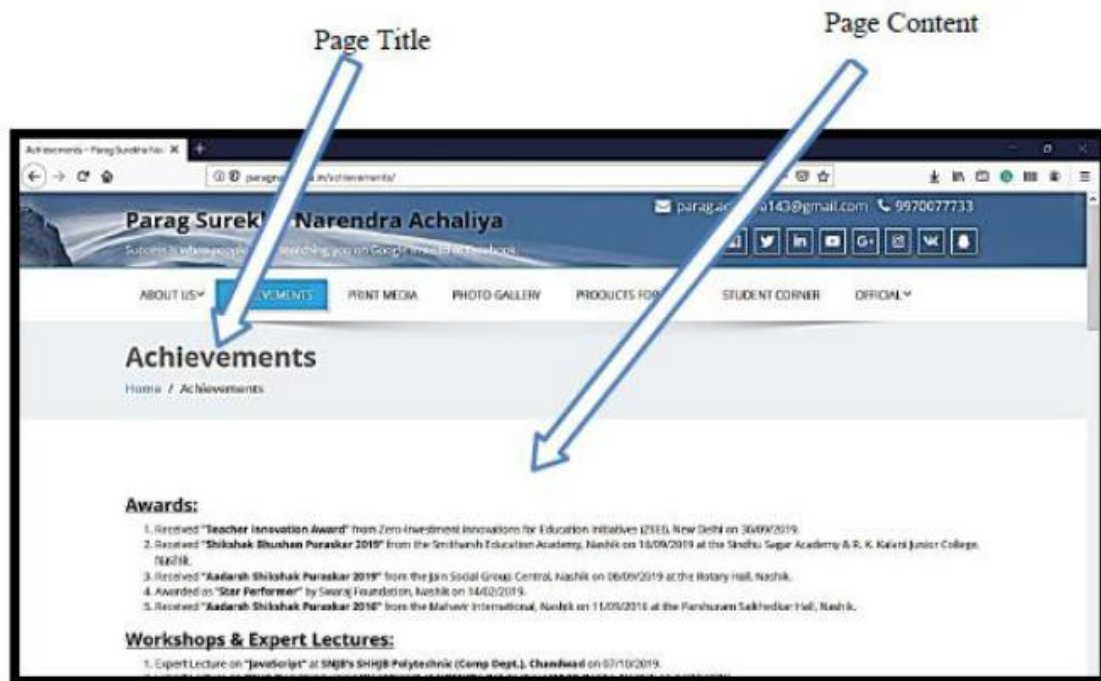
Header

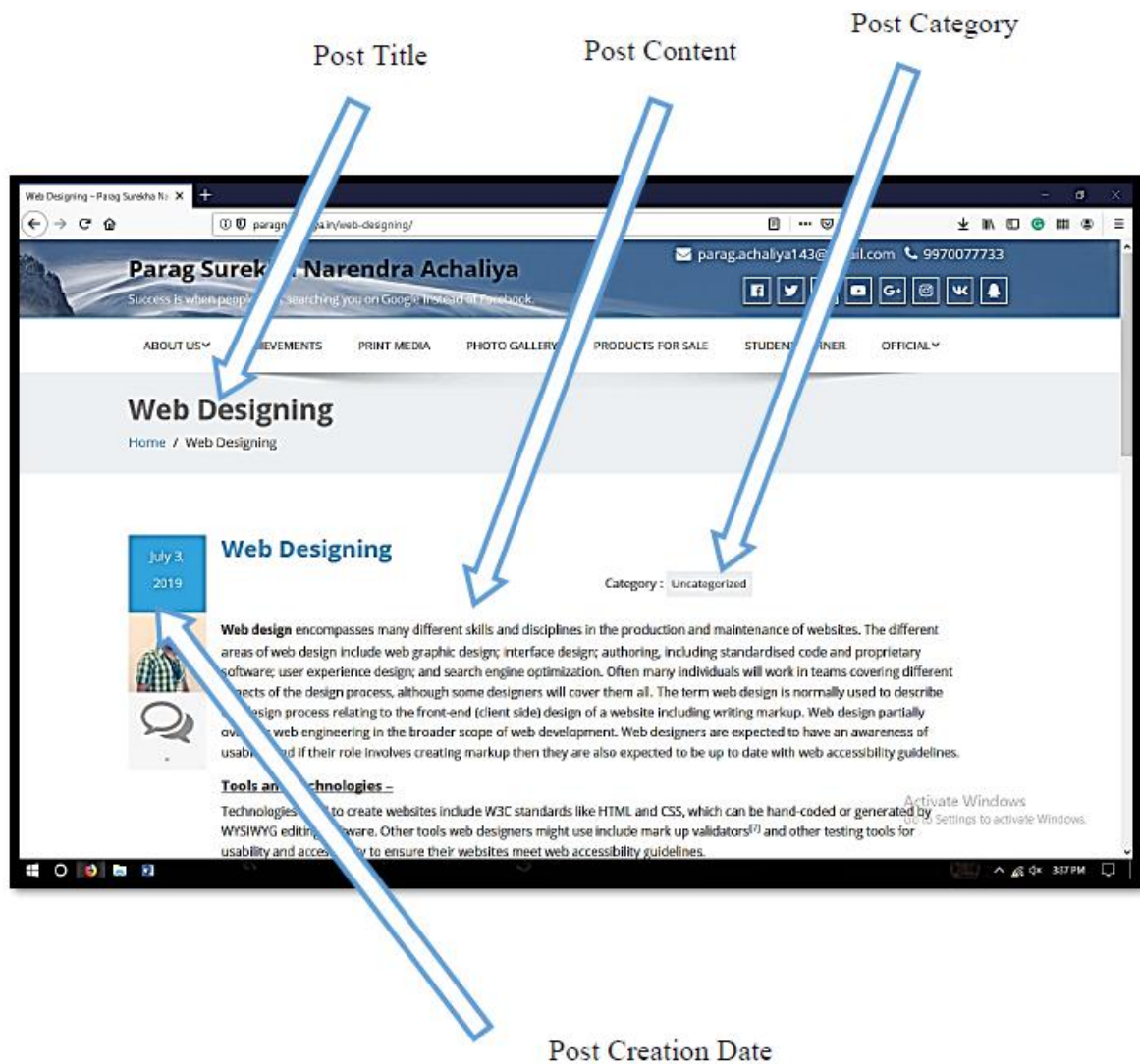
Tab

Menu



Slider





Exercise:

1. What is WordPress? How safe is a website on WordPress?
2. What are the positive aspects of WordPress? Are there any limitations to a WordPress website?
3. What are the disadvantages of WordPress?
4. What are the rules that you have to follow for WordPress plugin development?
5. What is the prefix of WordPress tables by default? How many tables are there in WordPress by default?
6. Why does WordPress use MySQL?
7. Why is a static front page used in WordPress and how can you create one?

8. What are the differences between Posts and Pages?
9. How to embed videos in WordPress?
10. Why is wordpress.com considered more secure than wordpress.org?
11. What are the system requirements for installing WordPress?
12. What are the steps should be followed for installing WordPress?
13. Explain the components shown on the Home screen of WordPress.
14. How many types of users WordPress have?
15. What steps you would take if a WordPress site is hacked?