

# PHP PROGRAMMING

---

**Internet:** It is a collection of interconnected networks

Or

A means of connecting a computer to any other computer anywhere in the world via dedicated routers and servers. When two computers are connected over the Internet, they can send and receive all kinds of information such as text, graphics, voice, video, and computer programs. Internet uses TCP/IP to transmit data via various types of media.

**World Wide Web (WWW):**The World Wide Web (WWW) is a network of online content that is formatted in HTML and accessed via HTTP. The term refers to all the interlinked HTML pages that can be accessed over the Internet. The World Wide Web was originally designed in 1991 by Tim Berners-Lee

**Website:** A site or website is a central location of various web pages that are all related and can be accessed by visiting the home page of the website using a browser.

**webpage:** It is a hypertext document connected to the World Wide Web.

Or

A web page or webpage is a document commonly written in HTML (Hypertext Markup Language) that is accessible through the Internet or other networks using an Internet browser. A web page is accessed by entering a URL address and may contain text, graphics, and hyperlinks to other web pages and files.

Or

A Web page is a document for the World Wide Web that is identified by a unique uniform resource locator (URL). A Web page can be accessed and displayed on a monitor or mobile device through a Web browser. The data found in a Web page is

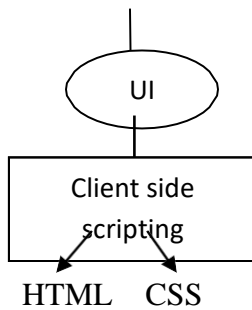
## **PHP PROGRAMMING**

---

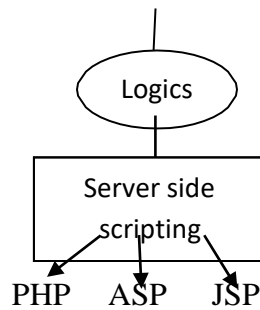
usually in HTML or XHTML format.

❖ Every web page is combination of

a) Designing



b) Coding

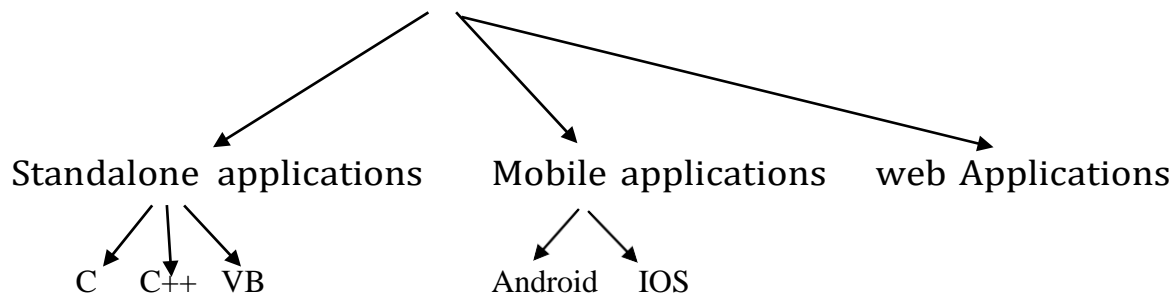


❖ Types of web pages

1. Static Web Pages – Does not contain logic
2. Dynamic Web Pages - Contains logic

**Application:** Application (app for short) is software designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user.

❖ We develop three types of applications.



Standalone applications are traditional software that are installed on each client system. For example(Turboc), in an office with 30 systems, standalone application needs to be installed on each of these 30 systems

**Web server:** Web servers are computers that deliver (serves up) Web pages. Every Web server has an IP address and domain name. For example, if you enter the URL <http://www.webopedia.com/index.html> in your browser, this sends a request to the Web server whose domain

# PHP PROGRAMMING

---

name is webopedia.com. The server then fetches the page named index.html and sends it to your browser. Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet.

## INTRODUCTION

PHP is server side scripting language implemented by Rasmus Lerdorf in 1994 using C and PERL languages. He implemented PHP 1.0 to track total number of visitors in his server. PHP stands for Personal Home Page. It also has an alias name Hypertext Preprocessor.

### What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

### What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

### What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database

## PHP PROGRAMMING

---

- PHP can be used to control user-access
- PHP can encrypt data

### Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: [www.php.net](http://www.php.net)
- PHP is easy to learn and runs efficiently on the server side
- This language is very simple to learn and runs efficiently on the server side.
- It supports many databases such as MySQL, Oracle, PostgreSQL etc.
- It is perfectly suited for Web development and can be embedded directly into the HTML code.
- PHP can also be used to create dynamic web pages.

**Features of PHP:** The main features of php is; it is a server side open source scripting language so you can free download this and use It is most popular and frequently used world wide scripting language, the main reason of popularity is; It is open source and very simple.

**Simple:** It is very simple and easy to use, compare to other scripting language, this is widely used all over the world

**Faster:** It is faster than other scripting language e.g. asp and jsp.

## PHP PROGRAMMING

---

**Interpreted:** It is an interpreted language, i.e. there is no need for compilation.

**Open Source:** Open source means you no need to pay for use php, you can free download and use.

**Case Sensitive:** PHP is case sensitive scripting language at time of variable declaration. In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive

**Simplicity:** PHP provides a lot of pre-define functions to secure your data. It is also compatible with many third-party applications, and PHP can easily integrate with other. In PHP script there is no need to include libraries like c, special compilation directives like Java, PHP engine starts execution from (<?) escape sequence and end with a closing escape sequence (<?). In PHP script, there is no need to write main function. And also you can work with PHP without creating a class.

**Efficiency:** PHP 4.0 introduced resource allocation mechanisms and more pronounced support for object-oriented programming, in addition to session management features. Eliminating unnecessary memory allocation.

**Platform Independent:** PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.

**Cross Database:** It is integrated with a number of popular databases, including MySQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

## PHP PROGRAMMING

---

**Security:** It supports different types of security functionalities to apply security to applications like one-way encryption, two-way encryption, authentication etc.

**Flexibility:** You can say that PHP is a very flexible language because of PHP is an embedded language you can embed PHP scripts with HTML, JAVA SCRIPT, WML, XML, and many others. You can run your PHP script any device like mobile Phone, tabs, laptops, PC and other because of PHP script execute on the server then after sending to the browser of your device.

**Familiarity:** If you are in programming background then you can easily understand the PHP syntax. And you can write PHP script because of most of PHP syntax inherited from other languages like C or Pascal.

**Error Reporting:** PHP have some predefined error reporting constants to generate a warning or error notice.

**Loosely Typed Language:** PHP supports variable usage without declaring its data type. It will be taken at the time of the execution based on the type of data it has on its value.

**Real-Time Access Monitoring:** PHP provides access logging by creating the summary of recent accesses for the user.

**Protocol Independent:** PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

**Cross Web Server:** It supports many web servers like IIS, Apache, Tomcat, etc.

# PHP PROGRAMMING

---

**Editor independent:** It supports different types of editors to develop programs or applications. We can also use light weight editors like Notepad, Edit+, etc.

## COMMON USES OF PHP

- It is used for create dynamic website.
- To Interacting with web server (Apache etc)
- To interacting with any back-end / database server e.g. MySQL
- To implement the business logical layers (one or more)
- Access Cookies variable and set cookies
- Using php you can restrict user to access specific web page
- PHP usually used to output HTML code to the browser
- It is used for send and receive E-Mails.
- You can use PHP to find today's date, and then build a calendar for the month.
- Using php you can count your visitors on your website.
- You can use PHP to create a special area of your website for members.
- Using php you can create login page for your user.
- Using php you can add, delete, modify elements within your database thru PHP.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.
- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- It can handle forms, i.e. gather data from files, save data to a file.

## VERSIONS OF PHP



# PHP PROGRAMMING

---

## PHP 1.0 (1994-95)

- Not a server side script
- Named as Personal Home Page

## PHP 2.0 (1997)

- Partially server side script
- Cross database

## PHP 3.0 (1998)

- Fully server side script
- Cross plat form
- Named as Hypertext Preprocessor

## PHP 4.0 (2000)

- Cross web server
- Zend engine 1.0 is introduced i.e., runtime engine of PHP

## PHP 5.0 (2004-05)

- Supports OOP concepts
- Zend 2.0 is introduced

## PHP 6.0 (2005)

- Added Unicode support

## PHP 7.0 (2015)

- XML and Web services support is increased
- Zend 3.0 is introduced
- Reduced memory usage

## **"Hello World" Script in PHP"**

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

<html>

```
<head>
    <title>Hello World</title>
</head>
<body>
    <?php echo "Hello, World!";?>
</body>
</html>
```

It will produce following result –

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags, are recognized by the PHP Parser.

**<?php PHP code goes here ?>**

**<? PHP code goes here ?>**

**<% PHP code goes here %>**

**<script language = "php"> PHP code goes here </script>**

A most common tag is the <?php...?> .

### **PHP is whitespace insensitive**

Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).

## PHP PROGRAMMING

---

PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row. one whitespace character is the same as many such characters.

For example, each of the following PHP statements that assigns the sum of 2 + 2 to the variable \$four is equivalent –

```
$four = 2 + 2; // single spaces
```

```
$four <tab>=<tab>2<tab>+<tab>2 ; // spaces and tabs
```

```
$four =
```

```
2+
```

```
2; // multiple lines
```

**PHP is case sensitive: PHP is a case sensitive language. Try out following example**

```
<?php
```

```
    $capital = 67;
```

```
    print("Variable capital is $capital<br>");
```

```
    print("Variable CaPiTaL is $CaPiTaL<br>");
```

```
?>
```

**Output:**

Variable capital is 67

Variable CaPiTaL is

**Running PHP Script from Command Prompt**

Yes you can run your PHP script on your command prompt. Assuming you have following content in test.php file

```
<?php
```

```
    echo "Hello PHP!!!!!!";
```

```
?>
```

Now run this script as command prompt as follows –

# PHP PROGRAMMING

---

\$ php test.php

It will produce the following result – Hello PHP!!!!

## IMPORTANT POINTS:

1. Every statement in PHP should be terminated by ';'.
2. PHP file extension should be .php.
3. PHP script should place within the script declaration tags.

```
<?php
```

```
.....
```

```
.....
```

```
?>
```

## SOFTWARES REQUIRED FOR TO WORK WITH PHP:

1. Browser - Internet Explorer
2. Server - Apache
3. Server side script - PHP
4. Database - MYSQL

## TOOLS TO WORK WITH PHP:

1. XAMPP - Cross Apache MYSQL PHP Perl
2. WAMPP - Window Apache MYSQL PHP Perl
3. LAMPP - Linux Apache MYSQL PHP Perl
4. MAMPP - Macintosh Apache MYSQL PHP Perl

## HOW TO DOWNLOAD & INSTALL XAMPP TOOL?

1. Go to Google
2. XAMPP download in Google Search
3. Select appropriate version of XAMPP based on OS
4. Click on download

## INSTALLATION

5. Double click on downloaded file

6. Click on the Next button
7. Select drive where you want to installed
8. Click on the Next button until you will get finished button
9. Click on Finish button

### **HOW TO START OR STOP APACHE SERVER?**

If you would like to run PHP program Apache server should be in start mode. To start or stop Apache server we have to follow below steps

1. Open XAMPP folder where we installed
2. Select Icon with name called XAMPP Hyper control
3. Double click on the Icon
4. It will open the window, click on start button related to Apache module.

### **HOW TO CHECK XAMPP S/W ARE SUCCESSFULLY RUNNING OR NOT?**

1. Open any browser
2. Type `http://localhost` in browser url address and press enter
3. If we can see welcome page XAMPP i.e., running successfully, so that we can start working with PHP

### **HOW TO WRITE PHP PROGRAM**

1. Open ant Editor
2. Create a New file
3. Save the file name with extension .php inside “htdocs” folder  
C:/XAMPP/htdocs(Ex: abc.php)
4. We need to write PHP code inside saved file  
Ex: `<?php`

.....

.....

?>

5. Save again php file

### HOW TO EXECUTE PHP PROGRAM

1. Open any browser
2. Type in the url `http://localhost/program.php` name and press enter
3. You can see now output if we don't have any error

#### Example:

1. Open notepad or any editor and implement PHP script.

```
<?php
    print "WELCOME";
    print "<br>";
    $x=100;
    print $x;
?>
```

2. Save this file in htdocs folder with .php extension.
3. To display output, open browser and enter URL address by providing site name with server name.

**TYPES OF ERRORS:** Mistake or fault occurs in a program is called error. There are basically 4 types of errors in PHP:

1. Notice
2. Warning
3. Fatal Error
4. Parse Error

#### 1. NOTICE:

---

## PHP PROGRAMMING

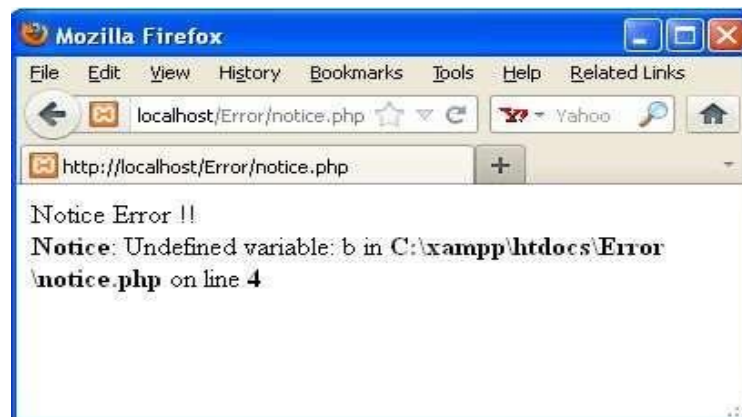
---

Notice that an error is the same as a warning error i.e. in the notice error does not stop execution of the script. **Notice that the error occurs when you try to access the undefined variable, and then produce a notice error.**

### Example

```
<?php
    $a="Uday ";
    echo "Notice Error !!";
    echo $b;
?>
```

**Output:** In the above code we defined a variable which named \$a. But we call another variable i.e. \$b, which is not defined. So there will be a notice error produced but execution of the script does not stop, you will see a message Notice Error !!. Like in the following image:



## 2. WARNING

Warning errors will not stop execution of the script. **The main reason for warning errors are to include a missing file or using the incorrect number of parameters in a function.**

### Example

```
<?php
    echo "Warning Error!!";
    include ("Welcome.php");
?>
```

**Output:** In the above code we include a welcome.php file, however the welcome.php file does not exist in the directory. So there will be a warning error produced but that does not stop the execution of the script i.e. you will see a message Warning Error !!.



### 3. FATAL ERROR

Fatal errors are caused when PHP understands what you've written; however what you're asking it to do can't be done. Fatal errors stop the execution of the script. **If you are trying to access the undefined functions, then the output is a fatal error.**

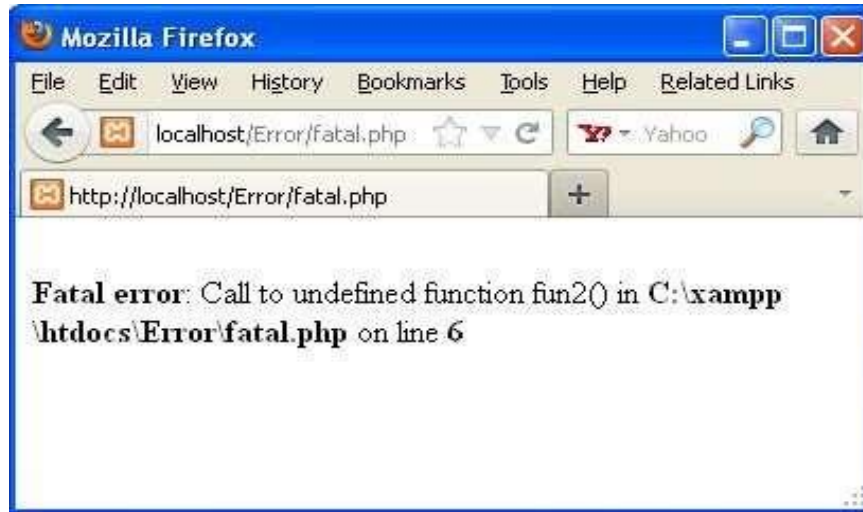
#### Example

```
<?php
    function fun1(){
        echo "Uday Kumar";
    }
```



```
fun2();  
echo "Fatal Error !!";  
?>
```

**Output:** In the above code we defined a function fun1 but we call another function fun2 i.e. func2 is not defined. So a fatal error will be produced that stops the execution of the script. Like as in the following image.



#### 4. PARSE ERROR

The parse error occurs if there is a syntax mistake in the script; the output is Parse errors. A parse error stops the execution of the script. There are many reasons for the occurrence of parse errors in PHP. The common reasons for parse errors are as follows:

Common reason of syntax errors are:

- a. Unclosed quotes
- b. Missing or Extra parentheses
- c. Unclosed braces
- d. Missing semicolon

#### Example

```
<?php
    echo "C#";
    echo "PHP"
    echo "C";
?>
```

**Output:** In the above code we missed the semicolon in the second line. When that happens there will be a parse or syntax error which stops execution of the script, as in the following image:



### INCLUDE MECHANISM IN PHP:

It is nothing but calling a file into another file. Whenever you have a requirement to call same piece of code which is available inside another page into required file then we have to go with include mechanism.

### TYPES OF INCLUDE MECHANISMS:

There are four types of functions available to achieve include mechanisms.

---

## PHP PROGRAMMING

---

**include():** The include() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.

Then create a file **menu.php** with the following content.

```
<a href="http://www.tutorialspoint.com/index.htm">Home</a>
<a href="http://www.tutorialspoint.com/ebxml">ebXML</a>
<a href="http://www.tutorialspoint.com/ajax">AJAX</a>
<a href="http://www.tutorialspoint.com/perl">PERL</a> <br />
```

Now create as many pages as you like and include this file to create header. For example now your test.php file can have following content.

```
<html>
<body>
    <?php include("menu.php"); ?>
    <p>This is an example to show how to include PHP
file!</p>
</body>
</html>
```

**Output:**



**require():** The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is

---

## PHP PROGRAMMING

---

any problem in loading a file then the `require()` function generates a fatal error and halt the execution of the script.

So there is no difference in `require()` and `include()` except they handle error conditions. It is recommended to use the `require()` function instead of `include()`, because scripts should not continue executing if files are missing or misnamed.

You can try using above example with `require()` function and it will generate same result. But if you will try following two examples where file does not exist then you will get different results.

```
<html>
  <body>
    <?php include("xxmenu.php"); ?>
    <p>This is an example to show how to include wrong PHP
file!</p>
  </body>
</html>
```

It will produce the result –This is an example to show how to include wrong PHP file!.Now lets try same example with `require()` function.

```
<html>
  <body>
    <?php require("xxmenu.php"); ?>
    <p>This is an example to show how to include wrong PHP
      file!</p>
  </body>
</html>
```

This time file execution halts and nothing is displayed.

**include\_once() & require\_once():** The `include_once()` and `require_once()` statements will only include the file once even if asked to include it a second time i.e. if the specified file has already been included in a previous statement, the file is not included again.

**Example:** x.php

```
<?php
    echo "GEEKSFORGEEKS";
?>
<?php
    include_once('header.inc.php');
    include_once('header.inc.php');
?>
```

**Output:** GEEKSFORGEEKS

```
<?php
    require_once('header.inc.php');
    require_once('header.inc.php');
?>
```

**Output:** GEEKSFORGEEKS

### TAGS IN PHP:

There are four different pairs of opening and closing tags which can be used in php. Here is the list of tags.

**1. Default syntax Or Universal Tags:** The default syntax starts with "<? php" and ends with "?>".

**Example:**

```
<?php
    echo "Default Syntax";
?>
```

**2. Short open Tags:** The short tags starts with "<?" and ends with "?>". Short style tags are only available when they are enabled in `php.ini` configuration file on servers.

**Example:**

```
<?
    echo "PHP example with short-tags";
?>
```

**3. HTML Script Tags:** HTML script tags look like this :

```
<script language="php">
    echo "This is HTML script tags.";
</script>
```

**4. ASP Style Tags:** The ASP style tags start with "<%" and ends with "%>". ASP style tags are only available when they are enabled in php.ini configuration file on servers.

**Example:**

```
<%
    echo 'This is ASP like style';
%>
```

**Note:** The above two tags and examples are given only for reference, but not used in practice any more.

### OUTPUT FUNCTIONS IN PHP:

#### 1. echo statement

- ❖ In PHP 'echo' statement is a language construct and not a function, so it can be used without parenthesis.
- ❖ But we are allowed to use parenthesis with echo statement when we are using more than one argument with it.
- ❖ The end of echo statement is identified by the semi-colon (;). We can use 'echo' to output strings or variables.
- ❖ Displaying Strings: The keyword echo followed by the string to be displayed within quotes.

```
<?php
```

```
    echo "Hello,This is a display string example!";  
?>
```

**Output:** Hello,This is a display string example!

- ❖ Displaying Strings as multiple arguments: We can pass multiple string arguments to the echo statement instead of single string argument, separating them by comma (',') operator. For example, if we have two strings say “Hello” and “World” then we can pass them as (“Hello”,“World”).

```
<?php  
    echo "Multiple ","argument ","string!";  
?>
```

**Output:** Multiple argument string!

- ❖ Displaying Variables: Displaying variables with echo statement is also as easy as displaying normal strings.

```
<?php  
    $text = "Hello, World!";  
    $num1 = 10;  
    $num2 = 20;  
    echo $text."\n";  
    echo $num1."+".$num2."=";  
    echo $num1 + $num2;  
?>
```

**Output:** Hello, World!

10+20=30

The (.) operator in the above code can be used to concatenate two strings in PHP and the “\n” is used for a new line and is also known as line-break.

### 2. print statement

- ❖ The PHP print statement is similar to the echo statement and can be used alternative to echo at many times.
- ❖ It is also language construct and so we may not use parenthesis: print or print ().
- ❖ The main difference between the print and echo statement is that print statement can have only one argument at a time and thus can print a single string. Also, print statement always returns a value 1.
- ❖ Like echo, print statement can also be used to print strings and variables.
- ❖ Displaying String of Text: We can display strings with print statement in the same way we did with echo statements. The only difference is we can not display multiple strings separated by comma(,) with a single print statement. Below example shows how to display strings with the help of PHP print statement:-

```
<?php
    print "Hello, world!";
?>
```

**Output:** Hello, world!

- ❖ Displaying Variables: Displaying variables with print statement is also same as that of echo statement.

```
<?php
    $text = "Hello, World!";
    $num1 = 10;
    $num2 = 20;
    print $text."\n";
```



## PHP PROGRAMMING

---

```
print $num1."+".$num2."=";  
print $num1 + $num2;
```

?>

**Output:** Hello, World!

10+20=30

Comparison between Echo and Print in PHP:

	"echo"	"print"
type	This is a type of output string.	This is a type of output string.
parenthesis	Not written with parenthesis.	It can or can not be written with parenthesis.
strings	It can output one or more strings.	It can output only one string at a time, and returns 1.
Functionality	echo is faster than print.	print is slower than echo.

### 3. var\_dump

Using this function, we can display the value of a variable along with data type.

Ex:

```
<?php
```

```
$x=100;  
$y="scott";  
var_dump($x);  
var_dump($y);
```

?>

**Output:**                int 100  
                          string(5) scott

4. **printf()** :Using this function, we can print variables with the help of format specifiers.

**Example:**

```
<?php
    $x=100;
    $y="scott";
    printf("%d",$x);
    printf("%s",$y);
?>
```

**Output:**        100   scott

5. **print\_r()**: Using this function, we can display all elements of array and properties of object.

**Example:**

```
<?php
    $x=array(10,20,30);
    print_r($x);
?>
```

**Output:**

```
Array
(
    [0]→10
    [1]→20
    [2]→30
)
```

## VARIABLES:

- ❖ Variable is an identifier which holds data or another one variable and whose value can be changed at the execution time of script.
- ❖ **Syntax:**      **\$variablename=value;**
- ❖ A variable starts with the \$ sign, followed by the name of the variable
- ❖ A variable name must start with a letter or the underscore character
- ❖ A variable name can't start with a number.
- ❖ A variable name can only contain alpha-numeric characters and underscores(A-z, 0-9 and \_ )
- ❖ Variable names are case-sensitive (\$str and \$STR both are two different)

## Example:

```
<?php
    $str="Hello world!";
    $a=5;
    $b=10.5;
    echo "String is: $str <br/>";
    echo "Integer is: $x <br/>";
    echo "Float is: $y <br/>";
?>
```

## VARIABLE SCOPES

- ❖ Scope of a variable is defined as its extent in program within which it can be accessed, i.e. the scope of a variable is the portion of the program with in which it is visible or can be accessed.
- ❖ Depending on the scopes, PHP has three variable scopes:

- 1. Local variables:** The variables declared within a function are called local variables to that function and has its scope only in that particular function. In simple words it cannot be accessed outside that function. Any declaration of a variable outside the function with same name as that of the one within the function is a complete different variable.

**Example:**

```
<?php
    $num = 60;
    function local_var()
    {
        $num = 50;
        echo "local num = $num \n";
    }
    local_var();
    echo "Variable num outside local_var() is $num \n";
?>
```

**Output:**    local num = 50  
                 Variable num outside local\_var() is 60

- 2. Global variables:** The variables declared outside a function are called global variables. These variables can be accessed directly outside a function. To get access within a function we need to use the “global” keyword before the variable to refer to the global variable.

**Example:**

```
<?php
    $num = 20;
```

```
function global_var()
{
    global $num;
    echo "Variable num inside function : $num \n";
}
global_var();
echo "Variable num outside function : $num \n";
?>
```

**Output:** Variable num inside function : 20

Variable num outside function : 20

**3. Static variable:** It is the characteristic of PHP to delete the variable, once it completes its execution and the memory is freed. But sometimes we need to store the variables even after the completion of function execution. To do this we use static keyword and the variables are then called as static variables.

**Example:**

```
<?php
function static_var()
{
    static $num = 5;
    $sum = 2;
    $sum++;
    $num++;
    echo $num "\n";
    echo $sum "\n";
}
static_var();
```

```
static_var();
```

```
?>
```

**Output:**    6     3

             7     3

### Example:

```
<?php
```

```
function myTest()
```

```
{
```

```
    static $x = 0;
```

```
    echo $x;
```

```
    $x++;
```

```
}
```

```
myTest();
```

```
myTest();
```

```
myTest();
```

```
?>
```

**Output:** 0 1 2

### CONSTANTS:

- ❖ Constants are name or identifier that can't be changed during the execution of the script. In php constants are define in two ways;

#### 1. Using define() function

#### 2. Using const keyword

- ❖ In php declare constants follow same rule variable declaration. Constant start with letter or underscore only.

- ❖ Create a PHP Constant

Create constant in php by using define() function.

**Syntax:**        define((name, value, case-insensitive)

**Name:** Specifies the name of the constant

**Value:** Specifies the value of the constant

**Case-insensitive:** Specifies whether the constant name should be case-insensitive. Default is false

**Example:**

```
<?php
    define("MSG","Hello world!");           //using case-sensitive
    echo MSG;

?>
```

**Output:** Hello world!

**Example:**

```
<?php
    define("MSG","Hello world!");           //using case-insensitive
    echo msg;

?>
```

**Output:** Hello world!

❖ Define constant using const keyword in PHP

The const keyword defines constants at compile time. It is a language construct not a function. It is bit faster than define(). It is always case sensitive.

**Example:**

```
<?php
    const MSG="Hello world!";
    echo MSG;

?>
```

## DATATYPES

## PHP PROGRAMMING

---

PHP data types are used to hold different types of data or values. PHP supports the following data types: String, Integer, Float, Boolean, Array, Object, NULL, and Resource.

### **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;
    $sum = $dec1 + $dec2;
    echo $sum;

?>
```

**Output:** 704

### **Float(or)double:**

- ❖ It 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

### String:

- ❖ It can 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

### 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
?>
```

### **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.
- ❖ 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  
              This condition is not TRUE

### **Arrays:**

- ❖ Array is a compound data-type which can store multiple values of same data type.

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

?>

**Output:** First Element: 10  
Second Element: 20  
Third Element: 30

### **Objects:**

- ❖ Objects are defined as instances of user defined classes that can hold both values and functions.

### **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.

### **COMMENTING PHP CODE**

- ❖ A comment is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP –

**1. Single-line comments** – They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

```
<?php
```

```
    # This is a comment, and
```

```
    # This is the second line of the comment
```

```
        // This is a comment too. Each style comments only
```

```
    print "An example with single line comments";
```

```
?>
```

**2. Multi-lines comments** – They are generally used to provide pseudo code algorithms and more detailed explanations when necessary. This style of commenting is the same as in C.

```
<?php
    /* This is a comment with multiline
       Subject: PHP
    */
    print "An example with multi line comments";
?>
```

**PHP Operators: Operators are used to perform operations on variables and values. PHP divides the operators in the following groups:**

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

**PHP Arithmetic Operators: The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of PHP's arithmetic operators:**

Operator	Description	Example	Result
+	Addition	\$x + \$y	Sum of \$x and \$y
-	Subtraction	\$x - \$y	Difference of \$x and \$y.

---

## PHP PROGRAMMING

---

*	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

```
<?php
$x = 10;
$y = 4;
echo($x + $y); // Outputs: 14
echo($x - $y); // Outputs: 6
echo($x * $y); // Outputs: 40
echo($x / $y); // Outputs: 2.5
echo($x % $y); // Outputs: 2
?>
```

**PHP Assignment Operators:** The assignment operators are used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	$\$x = \$y$	$\$x = \$y$
+=	Add and assign	$\$x += \$y$	$\$x = \$x + \$y$
-=	Subtract and assign	$\$x -= \$y$	$\$x = \$x - \$y$
*=	Multiply and assign	$\$x *= \$y$	$\$x = \$x * \$y$
/=	Divide and assign quotient	$\$x /= \$y$	$\$x = \$x / \$y$
%=	Divide and assign modulus	$\$x \% = \$y$	$\$x = \$x \% \$y$

```
<?php
$x = 10;
echo $x; // Outputs: 10
$x = 20;
$x += 30;
```

```
echo $x; // Outputs: 50
$x = 50;
$x -= 20;
echo $x; // Outputs: 30
$x = 5;
$x *= 25;
echo $x; // Outputs: 125
$x = 50;
$x /= 10;
echo $x; // Outputs: 5
$x = 100;
$x %= 15;
echo $x; // Outputs: 10
?>
```

**PHP Comparison Operators:** The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	<code>\$x == \$y</code>	True if \$x is equal to \$y
===	Identical	<code>\$x === \$y</code>	True if \$x is equal to \$y, and they are of the same type
!=	Not equal	<code>\$x != \$y</code>	True if \$x is not equal to \$y
<>	Not equal	<code>\$x &lt;&gt; \$y</code>	True if \$x is not equal to \$y
!==	Not identical	<code>\$x !== \$y</code>	True if \$x is not equal to \$y, or they are not of the same type
<	Less than	<code>\$x &lt; \$y</code>	True if \$x is less than \$y
>	Greater than	<code>\$x &gt; \$y</code>	True if \$x is greater than \$y

---

## PHP PROGRAMMING

---

>=	Greater than or equal to	\$x >= \$y	True if \$x is greater than or equal to \$y
<=	Less than or equal to	\$x <= \$y	True if \$x is less than or equal to \$y

```
<?php
```

```
$x = 25;
```

```
$y = 35;
```

```
$z = "25";
```

```
var_dump($x == $z); // Outputs: boolean true
```

```
var_dump($x === $z); // Outputs: boolean false
```

```
var_dump($x != $y); // Outputs: boolean true
```

```
var_dump($x !== $z); // Outputs: boolean true
```

```
var_dump($x < $y); // Outputs: boolean true
```

```
var_dump($x > $y); // Outputs: boolean false
```

```
var_dump($x <= $y); // Outputs: boolean true
```

```
var_dump($x >= $y); // Outputs: boolean false
```

```
?>
```

### PHP Incrementing and Decrementing Operators

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

Operator	Name	Effect
++\$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-      Returns \$x, then decrements \$x  
             decrement      by one

```
<?php
```

```
$x = 10;
```

```
echo ++$x; // Outputs: 11
```

```
echo $x;    // Outputs: 11
```

```
$x = 10;
```

```
echo $x++; // Outputs: 10
```

```
echo $x;    // Outputs: 11
```

```
$x = 10;
```

```
echo --$x; // Outputs: 9
```

```
echo $x;    // Outputs: 9
```

```
$x = 10;
```

```
echo $x--; // Outputs: 10
```

```
echo $x;    // Outputs: 9
```

```
?>
```

### PHP Logical Operators

The logical operators are typically 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



---

## PHP PROGRAMMING

---

```
!           Not      !$x           True if $x is not true

<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))){
    echo "$year is a leap year.";
}
else
{
    echo "$year is not a leap year.";
}
?>
```

### PHP String Operators

There are two operators which are specifically designed for strings.

Operator	Description	Example	Result
.	<b>Concatenation</b>	<b>\$str1 . \$str2</b>	<b>Concatenation of \$str1 and \$str2</b>
.=	<b>Concatenation</b>	<b>\$str1 .= \$str2</b>	<b>Appends the \$str2 to the \$str1 assignment</b>

```
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!
$x .= $y;
echo $x; // Outputs: Hello World!
?>
```

### PHP Array Operators

---

## PHP PROGRAMMING

---

The array operators are used to compare arrays:

Operator	Name	Example	Result
+	Union	<code>\$x + \$y</code>	Union of <code>\$x</code> and <code>\$y</code>
<code>==</code>	Equality	<code>\$x == \$y</code>	True if <code>\$x</code> and <code>\$y</code> have the same key/value pairs
<code>===</code>	Identity	<code>\$x === \$y</code>	True if <code>\$x</code> and <code>\$y</code> have the same key/value pairs in the same order and of the same types
<code>!=</code>	Inequality	<code>\$x != \$y</code>	True if <code>\$x</code> is not equal to <code>\$y</code>
<code>&lt;&gt;</code>	Inequality	<code>\$x &lt;&gt; \$y</code>	True if <code>\$x</code> is not equal to <code>\$y</code>
<code>!==</code>	Non-identity	<code>\$x !== \$y</code>	True if <code>\$x</code> is not identical to <code>\$y</code>

```
<?php
```

```
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
```

```
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
```

```
$z = $x + $y; // Union of $x and $y
```

```
var_dump($z);
```

```
var_dump($x == $y); // Outputs: boolean false
```

```
var_dump($x === $y); // Outputs: boolean false
```

```
var_dump($x != $y); // Outputs: boolean true
```

```
var_dump($x <> $y); // Outputs: boolean true
```

```
var_dump($x !== $y); // Outputs: boolean true
```

```
?>
```

### Ternary operator:

```
<?php
```

```
    $a = 10;
```

```
    $b = 20;
```

```
/* If condition is true then assign a to result otheriwse b */
$result = ($a > $b ) ? $a :$b;

echo "TEST1 : Value of result is $result<br/>";

/* If condition is true then assign a to result otheriwse b */
$result = ($a < $b ) ? $a :$b;

echo "TEST2 : Value of result is $result<br/>";

?>
```

### DECISION MAKING

- ❖ PHP allows us to perform actions based on some type of conditions that may be logical or comparative.
- ❖ Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed.
- ❖ PHP provides us with four conditional statements:
  1. if statement
  2. if...else statement
  3. if...elseif...else statement
  4. switch statement

**1. if Statement:** The *if* statement is used to execute a block of code only if the specified condition evaluates to true. This is the simplest PHP's conditional statements and can be written like:

#### Syntax :

```
if (condition)
```

```
{  
    // if TRUE then execute this code  
}
```

**Example:**

```
<?php  
    $x = 12;  
    if ($x > 0)  
    {  
        echo "The number is positive";  
    }  
?>
```

**Output:** The number is positive

**2. if...else Statement:** You can enhance the decision making process by providing an alternative choice through adding an *else* statement to the *if* statement. The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false. It can be written, like this:

**Syntax:**

```
if (condition)  
{  
    // if TRUE then execute this code  
}  
else  
{  
    // if FALSE then execute this code  
}
```

**Example:**

```
<?php
    $x = -12;
    if ($x > 0)
        echo "The number is positive";
    else
        echo "The number is negative";
?>
```

**Output:** The number is negative

**3. if...elseif...else Statement:** he *if...elseif...else* a special statement that is used to combine multiple *if...else* statements.. We use this when there are multiple conditions of TRUE cases.

**Syntax:**

```
if (condition)
{
    // if TRUE then execute this code
}
elseif
{
    // if TRUE then execute this code
}
elseif
{
    // if TRUE then execute this code
}
else
{
```

```
        // if FALSE then execute this code
    }
```

**Example:**

```
<?php
    $x = "August";
    if ($x == "January")
    {
        echo "Happy Republic Day";
    }
    elseif ($x == "August")
    {
        echo "Happy Independence Day!!!";
    }
    else
    {
        echo "Nothing to show";
    }
?>
```

**Output:** Happy Independence Day!!!

**4. switch Statement:** The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, break and default.

The break statement is used to stop the automatic control flow into the next cases and exit from the switch case.

The default statement contains the code that would execute if none of the cases match.

**Syntax:**

```
switch(expression)
{
    case value1:
        code to be executed if n==statement1;
        break;
    case value 2:
        code to be executed if n==statement2;
        break;
    case value 3:
        code to be executed if n==statement3;
        break;
    case value 4:
        code to be executed if n==statement4;
        break;
    .....
    default:
        code to be executed if n != any case;
}
```

**Example:**

```
<?php
    $n = "February";
    switch($n)
```

```
{
    case "January": echo "Its January";
        break;
    case "February": echo "Its February";
        break;
    case "March": echo "Its March";
        break;
    case "April": echo "Its April";
        break;
    case "May": echo "Its May";
        break;
    default: echo "Doesn't exist";
}
```

?>

**Output:** Its February

### LOOPS

❖ Loops are used to execute the same block of code again and again, until a certain condition is met. The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort. PHP supports four different types of loops.

1. for loop
2. while loop
3. do-while loop
4. foreach loop

**1. for loop:** This type of loops is used when the user knows in advance, how many times the block needs to execute. These type of loops are also known as entry-controlled loops. There are three



main parameters to the code, namely the initialization, the test condition and the counter.

**Syntax:**

```
for (initialization expression; test condition; update expression)
{
    // code to be executed
}
```

In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than counter value. If statement is true, then loop body is executed and loop variable gets updated. Steps are repeated till exit condition comes.

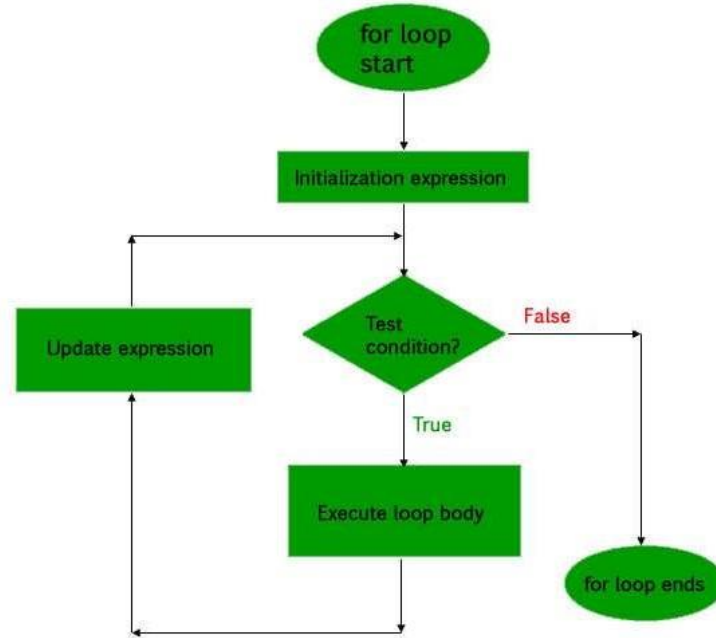
**Example:**

```
<?php
    for ($num = 1; $num <= 10; $num += 2)
    {
        echo "$num \n";
    }
?>
```

**Output:**

1    3    5    7    9

**Flow Diagram:**



**2. while loop:** The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.

**Syntax:**

```
while (if the condition is true)
{
    // code is executed
}
```

**Example:**

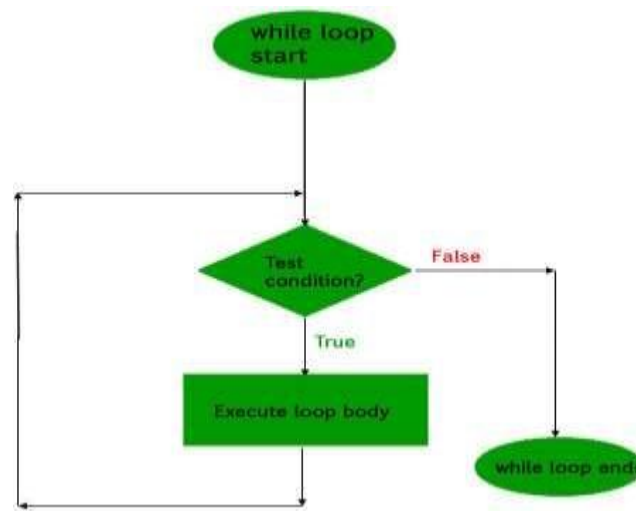
```
<?php
$num = 2;
while ($num < 12)
{
    $num += 2;
```

```
    echo $num, "\n";  
}
```

?>

**Output:** 4      6      8      10      12

**Flowchart**



**3. do-while loop:** This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do...while loop. After executing once, the program is executed as long as the condition holds true.

**Syntax:**

```
do  
{  
    //code is executed  
} while (if condition is true);
```

**Example:**

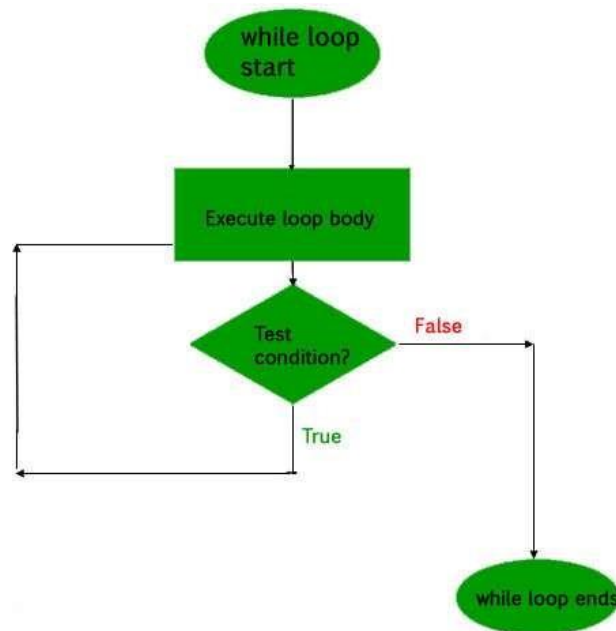
<?php

```
$num = 2;  
do {  
    $num += 2;  
    echo $num, "\n";  
} while ($num < 12);
```

?>

**Output:** 4      6      8      10      12

**Flowchart:**



**4. foreach loop:** The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to \$value and the array pointer is moved by one and in the next pass next element will be processed.

**Syntax:**

```
foreach (array_element as value)  
{
```

```
        //code to be executed
    }
```

### Example:

```
<?php
    $arr = array (10, 20, 30, 40, 50, 60);
    foreach ($arr as $value)
    {
        echo "$val \n";
    }

    $arr = array ("Ram", "Laxman", "Sita");
    foreach ($arr as $value)
    {
        echo "$val \n";
    }
?>
```

**Output:**    10    20    30    40    50    60  
              Ram Laxman    Sita

### Break:

The PHP break keyword is used to terminate the execution of a loop prematurely. The break statement is placed inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

**Example:** In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<?php
```

```
$i = 0;
while( $i < 10)
{
    $i++;
    if( $i == 3 )
        break;
}
echo ("Loop stopped at i = $i" );
?>
```

**Output:** Loop stopped at i = 3

### **Continue:**

The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop.

Just like the break statement the continue statement is placed inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

**Example:** In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<?php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value )
{
    if( $value == 3 )
        continue;
    echo "Value is $value <br />";
}
```

```
}  
?>
```

### **Output:**

Value is 1

Value is 2

Value is 4

Value is 5

### **swapping Using Third Variable**

```
<?php  
$a = 45;  
$b = 78;  
// Swapping Logic  
$third = $a;  
$a = $b;  
$b = $third;  
echo "After swapping:<br><br>";  
echo "a = ".$a." b=".$b;  
?>
```

### **Swapping Without using Third Variable(+ and -):**

```
<?php  
$a=234;  
$b=345;  
//using arithmetic operation  
$a=$a+$b;  
$b=$a-$b;  
$a=$a-$b;  
echo "Value of a: $a</br>";
```

```
echo "Value of b: $b</br>";
```

```
?>
```

**Example for (\* and /):**

```
<?php
```

```
$a=234;
```

```
$b=345;
```

```
// using arithmetic operation
```

```
$a=$a*$b;
```

```
$b=$a/$b;
```

```
$a=$a/$b;
```

```
echo "Value of a: $a</br>";
```

```
echo "Value of b: $b</br>";
```

```
?>
```

**PHP script for generating a list of prime numbers below 100.**

```
<?php
```

```
$number = 2 ;
```

```
while ($number < 100 )
```

```
{
```

```
    $div_count=0;
```

```
    for ( $i=1;$i<=$number;$i++)
```

```
    {
```

```
        if (($number%$i)==0)
```

```
        {
```

```
            $div_count++;
```

```
        }
```

```
    }
```

```
    if ($div_count<3)
```



```
{  
    echo $number." , ";  
}  
  
    $number=$number+1;  
}  
?>
```

**output:** 2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 ,  
47 , 53 , 59 , 61 , 67 , 71 , 73 , 79 , 83 , 89 , 97 ,

### **Factorial of a number**

```
<?php  
$num = 4;  
$factorial = 1;  
for ($x=$num; $x>=1; $x--)  
{  
    $factorial = $factorial * $x;  
}  
echo "Factorial of $num is $factorial";  
?>
```

### **Palindrome Number Program Without of Using PHP Predefined Function :**

```
<?php  
$number = 53235;  
$p = $number;  
$revnum =0;  
while($number != 0)  
{  
    $revnum = $revnum*10 + $number % 10 ;
```

```
$number = (int)($number/10);  
}  
if($revnum==$p)  
    echo $p.' is palindrome number';  
else  
    echo 'number is not palindrome';  
?>
```

### Reversing Number in PHP

#### Example:

```
<?php  
$num = 23456;  
$revnum = 0;  
while ($num > 1)  
{  
    $rem = $num % 10;  
    $revnum = ($revnum * 10) + $rem;  
    $num = ($num / 10);  
}  
echo "Reverse number of 23456 is: $revnum";  
?>
```

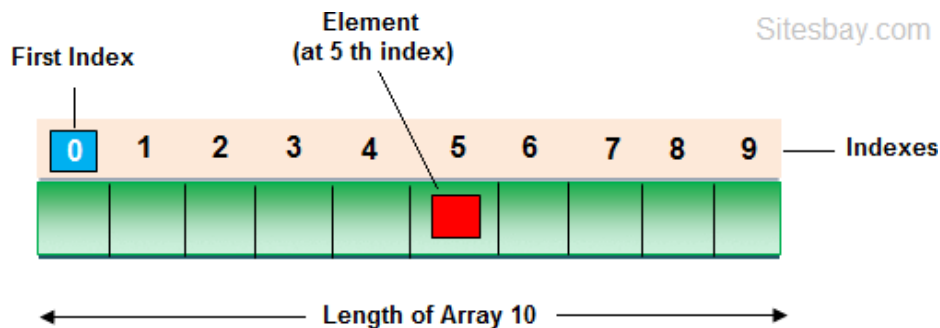
## UNIT II

**Arrays**-What is an Array, Creating an array, Accessing array Element, Types of arrays, array functions.

**Functions**-What is a function, Define a function, Call by value and Call by reference, Recursive functions

### ARRAYS

Array is used to store multiple values of same type in single variable. An array is a special variable, which can hold more than one value at a time.



### CREATE AN ARRAY IN PHP

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

**Syntax:** `array( );`

### TYPES OF ARRAY IN PHP

There are three types of array in PHP, which are given below.

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

#### 1. Indexed Arrays

The index can be assigned automatically (index always starts at 0).

#### Example

```
<?php
```

## PHP PROGRAMMING

---

```
$student = array("Harry", "Varsha", "Gaurav");  
echo "Class 10th Students " . $student[0] . ", " . $student[1] . "  
and " . $student[2] . ".";  
?>
```

**Output:** Class 10th Students Harry, Varsha and Gaurav

### Find Length of an Array in PHP

Using count() function you can find length of an array in php.

#### Example

```
<?php  
$student = array("Harry", "Varsha", "Gaurav");  
echo "Length of Array: ";  
echo count($student);  
?>
```

**Output:** Length of Array: 3

### Arrays using for Loop

#### Example

```
<?php  
$student = array("Harry", "Varsha", "Gaurav");  
$arrlength = count($student);  
for($i = 0; $i < $arrlength; $i++)  
{  
    echo $student[$i];  
    echo "<br>";  
}  
?>
```

**Output:** Harry  
Varsha

Gaurav

## 2. Associative Arrays in PHP

In this type of array; arrays use named keys that you assign to them.

### Syntax

```
$age = array("Harry"=>"10", "Varsha"=>"20", "Gaurav"=>"30");
```

or

```
$age['Harry'] = "10";
```

```
$age['Varsha'] = "20";
```

```
$age['Gaurav'] = "30";
```

## 3. Multidimensional Arrays in PHP

A multidimensional array is an array containing one or more arrays.

For a two-dimensional array you need two indices to select an element

### Example

```
<?php
```

```
    $student      =      array(          array("Harry",300,11),
    array("Varsha",400,10),
    array("Gaurav",200,8), array("Hitesh",220,8));
```

```
    echo  $student[0][0].":  Marks:  ".$student[0][1].",  Class:
    ".$student[0][2]."<br>;
```

```
    echo  $student[1][0].":  Marks:  ".$student[1][1].",  Class:
    ".$student[1][2]."<br>;
```

```
    echo  $student[2][0].":  Marks:  ".$student[2][1].",  Class:
    ".$student[2][2]."<br>;
```

```
    echo  $student[3][0].":  Marks:  ".$student[3][1].",  Class:
    ".$student[3][2]."<br>;
```

```
?>
```

**Output:** Harry: Marks: 300 Class: 11  
Varsha: Marks: 400 Class: 10  
Gaurav: Marks: 200 Class: 8  
Hitesh: Marks: 220 Class: 8

### **SORT FUNCTIONS FOR ARRAYS**

#### **1. sort()** - sort arrays in ascending order

##### **Example**

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

**Output:** BMW  
Toyota  
Volvo

#### **2. rsort()** - sort arrays in descending order

##### **Example**

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    rsort($cars);
    $clength = count($cars);
    for($x = 0; $x < $clength; $x++)
```

```
{  
    echo $cars[$x];  
    echo "<br>";  
}  
?>
```

**Output:** Volvo  
Toyota  
BMW

**3. asort()** - sort associative arrays in ascending order, according to the value

**Example:**

```
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
asort($age);  
foreach($age as $x => $x_value)  
{  
    echo "Key=" . $x . ", Value=" . $x_value;  
    echo "<br>";  
}  
?>
```

**Output:** Key=Peter, Value=35  
Key=Ben, Value=37  
Key=Joe, Value=43

**4. ksort()** - sort associative arrays in ascending order, according to the key

**Example:**

```
<?php
```

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
foreach($age as $x => $x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

**Output:**   Key=Ben, Value=37  
              Key=Joe, Value=43  
              Key=Peter, Value=35

**5. arsort()** - sort associative arrays in descending order, according to the value

**Example:**

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
foreach($age as $x => $x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

**Output:**   Key=Joe, Value=43  
              Key=Ben, Value=37  
              Key=Peter, Value=35



**6. krsort()** - sort associative arrays in descending order, according to the key

**Example:**

```
<?php
    $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
    krsort($age);
    foreach($age as $x => $x_value)
    {
        echo "Key=" . $x . ", Value=" . $x_value;
        echo "<br>";
    }
?>
```

**Output:**   Key=Peter, Value=35  
              Key=Joe, Value=43  
              Key=Ben, Value=37

**array\_change\_key\_case()**   Changes all keys in an array to lowercase or uppercase

**Syntax:**

```
array_change_key_case(array $array[, int $case = CASE_LOWER ] )
```

**Example**

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

**Output:**

Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )

**array\_chunk():** Splits an array into chunks of arrays

**Syntax:** array\_chunk ( array \$array , int \$size )

**Example**

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"2000
00");
print_r(array_chunk($salary,2));
?>
```

**Output:**

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

### ARRAY FUNCTIONS

Function	Description
array()	Creates an array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_assoc()	Compare arrays, and returns the differences (compare keys and values)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_fill()	Fills an array with values

## PHP PROGRAMMING

---

array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)

### Example

```
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
    echo "$n<br />";
}
?>
```

**Output:**   sonoo  
             smith

array_intersect_assoc()	Compare arrays and returns the matches (compare keys and values)
array_intersect_key()	Compare arrays, and returns the matches (compare keys only)
array_intersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function)

## PHP PROGRAMMING

---

<code>array_intersect_ukey()</code>	Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function)
<code>array_key_exists()</code>	Checks if the specified key exists in the array
<code>array_keys()</code>	Returns all the keys of an array
<code>array_map()</code>	Sends each value of an array to a user-made function, which returns new values
<code>array_merge()</code>	Merges one or more arrays into one array
<code>array_merge_recursive()</code>	Merges one or more arrays into one array recursively
<code>array_multisort()</code>	Sorts multiple or multi-dimensional arrays
<code>array_pad()</code>	Inserts a specified number of items, with a specified value, to an array
<code>array_pop()</code>	Deletes the last element of an array
<code>array_product()</code>	Calculates the product of the values in an array
<code>array_push()</code>	Inserts one or more elements to the end of an array
<code>array_rand()</code>	Returns one or more random keys from an array
<code>array_reduce()</code>	Returns an array as a string, using a user-defined function
<code>array_replace()</code>	Replaces the values of the first array with the values from following arrays
<code>array_replace_recursive()</code>	Replaces the values of the first array with the values from following arrays recursively
<code>array_reverse()</code>	Returns an array in the reverse order

### Example

```
<?php
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
    echo "$s<br />";
}
?>
```

**Output:** autumn  
spring  
winter  
summer

**array\_search():** Searches an array for a given value and returns the key

### Example

```
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

**Output:** 2

array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array

## PHP PROGRAMMING

---

`array_sum()` Returns the sum of the values in an array

**Example:**

```
<?php
$a = array(2, 4, 6, 8);
echo "sum(a) = " . array_sum($a) . "\n";
$b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
echo "sum(b) = " . array_sum($b) . "\n";
?>
```

**Output:**    `sum(a) = 20`  
              `sum(b) = 6.9`

`array_unique()` Removes duplicate values from an array

`array_unshift()` Adds one or more elements to the beginning of an array

`array_values()` Returns all the values of an array

`count()` Returns the number of elements in an array

`current()` Returns the current element in an array

`each()` Returns the current key and value pair from an array

`end()` Sets the internal pointer of an array to its last element

`extract()` Imports variables into the current symbol table from an array

`in_array()` Checks if a specified value exists in an array

`key()` Fetches a key from an array

`list()` Assigns variables as if they were an array

`range()` Creates an array containing a range of elements

## PHP PROGRAMMING

---

`reset()` Sets the internal pointer of an array to its first element

`shuffle()` Shuffles an array

`sizeof()` Alias of `count()`

`uksort()` Sorts an array by keys using a user-defined comparison function

`usort()` Sorts an array using a user-defined comparison function

**natsort()**:The `natsort()` function is used to sort an array using a "natural order" algorithm. The function implements a sort algorithm but maintains original keys/values.

This function implements a sort algorithm that orders alphanumeric strings in the way a human being would while maintaining key/value associations.

**Syntax:** `natsort(array_name)`

**Example:**

```
<?php
$php_files = array("code12.php", "code22.php",
"code2.php", "Code3.php", "code1.php");
natsort($php_files);
echo "List of file using natural order: ";
print_r($php_files);
?>
```

**Output:**List of file using natural order:

Array

```
(
    [3] => Code3.php
```

```
[4] => code1.php
[2] => code2.php
[0] => code12.php
[1] => code22.php
)
```

**natcasesort():**The natcasesort() function is used to sort an array using a case insensitive "natural array" algorithm. The function implements a sort algorithm but maintains original keys/values.

natcasesort() is a case insensitive version of natsort()

**Syntax:** natcasesort(array\_name)

**Example:**

```
<?php
$php_files = array("code12.php", "code22.php",
"code2.php", "Code3.php", "code1.php");
natcasesort($php_files);
echo "List of file using natural order: ";
print_r($php_files);
?>
```

**Output:**List of file using natural order:

```
Array
(
    [4] => code1.php
    [2] => code2.php
    [3] => Code3.php
    [0] => code12.php
    [1] => code22.php
)
```



### usort():

The usort() function sorts an array by a user defined comparison function. This function assigns new keys for the elements in the array. Existing keys will be removed.

### Syntax

```
usort ( $array, $cmp_function )
```

### Parameters

Sr.No	Parameter & Description
1	<b>array(Required)</b> It specifies an array.
2	<b>cmp_function(Required)</b> Useful defined function to compare values and to sort them. <ul style="list-style-type: none"><li>• If a = b, return 0</li><li>• If a &gt; b, return 1</li><li>• If a &lt; b, return -1</li></ul>

### Example

```
<?php
function cmp_function($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana" );
usort($fruits, "cmp_function");
print_r($fruits);
?>
```

**Output:**     Array ( [0] => orange [1] => lemon [2] => banana )

---

## PHP PROGRAMMING

---

**extract():** it converts array keys into variable names and array values into variable value.

**Syntax:**    **extract(array,extract\_rules,prefix)**

**Example:**

```
<?php
$state = array("AS"=>"ASSAM", "OR"=>"ORRISA", "KR"=>"KERELA");
extract($state);
echo" $AS is $AS\n $KR is $KR\n $OR is $OR";
?>
```

**Output:**    \$AS is ASSAM  
              \$KR is KERELA  
              \$OR is ORRISA

**Example:**

```
<?php
$AS="Original";
$state = array("AS"=>"ASSAM", "OR"=>"ORRISA", "KR"=>"KERELA");
extract($state, EXTR_PREFIX_SAME, "dup");
echo"\$AS is $AS\n $KR is $KR\n $OR if $OR \n $dup_AS =
$dup_AS";
?>
```

**Output:**    \$AS is Original  
              \$KR is KERELA  
              \$OR is ORRISA  
              \$dup\_AS = ASSAM

**current():** The current() function is used to fetch the value of the current element in an array.

**Syntax:**    **current(array\_name)**

**Example:**

```
<?php
$subject= array('Language','English','Math','Science');
$result = current($subject);
echo $result;
?>
```

**Output:** Language

**in\_array():** The in\_array() function is used to check whether a value exists in an array or not.

**Syntax:** in\_array(search\_value, array\_name, mode)

**Parameters:**

Name	Description	Required / Optional	Type
search_value	Value to search in the array.	Required	Mixed*
array_name	Specifies the array to search.	Required	Array
mode	If it is set to true, the function checks the type of the search_value.	Optional	Boolean

```
<?php
$number_list = array('16.10', '22.0', '33.45', '45.45');
if (in_array(22.0, $number_list))
{
echo "'22.0' found in the array";
}
?>
```

**Output:** '22.0' found in the array

**next():** The next() function is used to advance the internal array pointer. next() behaves like current(), with one difference. It advances the internal array pointer one place forward before returning the element value.

**Syntax:** next(array\_name)

**Example:**

```
<?php
$val1 = array('Language', 'Math', 'Science', 'Geography');
$cval = current($val1);
echo "$cval <br />";
$cval = next($val1);
echo "$cval <br />";
$cval = next($val1);
echo "$cval <br />";
$cval= prev($val1);
echo "$cval <br />";
$cval= end($val1);
echo "$cval <br />";
?>
```

**Output:** Language

Math

Science

Math

Geography

**Prev:** The prev() function is used to fetch the array value in the previous place, pointed by the internal array pointer.

prev() function behaves just like next(), except it rewinds the internal array pointer one place instead of advancing it.

**Syntax:** prev(array\_name)

```
<?php
$val1 = array('Language', 'Math', 'Science', 'Geography');
$cval = current($val1);
echo "$cval <br />";
$cval = next($val1);
echo "$cval <br />";
$cval = next($val1);
echo "$cval <br />";
$cval = prev($val1);
echo "$cval <br />";
?>
```

**Output:** Language  
Math  
Science  
Math

**range();** The range() function used to create an array containing a range of elements.

**Syntax:** range(low\_value, high\_value, step)

**Parameters:**

Name	Description	Required / Optional	Type
low_value	Specify the lowest value.	Required	Mixed*
high_value	Specify the highest value.	Required	Mixed*

---

## PHP PROGRAMMING

---

step	It is used as increment between elements. The default value is 1.	Optional	Integer/Float
------	--	----------	---------------

### Example:

```
<?php
$number_list = range(11,19);
print_r ($number_list);
echo "<br /> ";
$number_list_step = range(11,19,2);
print_r ($number_list_step);
echo "<br /> ";
$letter_list = range("u","z");
print_r ($letter_list);
?>
```

### Output:

```
Array ( [0] => 11 [1] => 12 [2] => 13 [3] => 14 [4] => 15 [5] => 16 [6] =>
17 [7] => 18 [8] => 19 )
Array ( [0] => 11 [1] => 13 [2] => 15 [3] => 17 [4] => 19)
Array ( [0] => u [1] => v [2] => w [3] => x [4] => y [5] => z)
```

**reset():** The reset() function rewinds array's internal pointer to the first element and returns the value of the first array element

**Syntax:**     **reset ( \$array );**

### Example

```
<?php
$input = array('foot', 'bike', 'car', 'plane');
$mode = end($input);
print "$mode <br />";
```

```
$mode = reset($input);  
print "$mode <br />";  
$mode = current($input);  
print "$mode <br />";  
?>
```

**Output:**   plane  
              foot  
              foot

**end():** The end() function advances array's internal pointer to the last element, and returns its value.

**Syntax:**    end ( \$array );

### Example

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

**Output:**    foot  
             bike  
             bike  
             foot  
             plane  
             plane

### FUNCTIONS

- ❖ A function is a block of code written in a program to perform some specific task. Functions take information's as parameter, execute a block of statements or perform operations on these parameters and return the result.
- ❖ PHP provides us with two major types of functions:
  - 1. Built-in functions:** PHP provides us with huge collection of built-in library functions. These functions are already coded and stored in form of functions. To use those we just need to call them as per our requirement like, `var_dump`, `fopen()`, `print_r()`, `gettype()` and so on.
  - 2. User Defined Functions:** Apart from the built-in functions, PHP allows us to create our own customized functions called the user-defined functions.
- ❖ Why should we use functions?
  - 1. Reusability:** If we have a common code that we would like to use at various parts of a program, we can simply contain it within a function and call it whenever required. This reduces the time and effort of repetition of a single code. This can be done both within a program and also by importing the PHP file, containing the function, in some other program



- 2. Easier error detection:** Since, our code is divided into functions, we can easily detect in which function, and the error could lie and fix them fast and easily.
- 3. Easily maintained:** If anything or any line of code needs to be changed, we can easily change it inside the function and the change will be reflected everywhere, where the function is called. Hence, easy to maintain.

### CREATING A FUNCTION

- ❖ While creating a user defined function we need to keep few things in mind:
1. Any name ending with an open and closed parenthesis is a function.
  2. A function name always begins with the keyword function.
  3. To call a function we just need to write its name followed by parenthesis.
  4. A function name cannot start with a number. It can start with an alphabet or underscore.
  5. A function name is not case-sensitive.

### Syntax:

```
function function_name()
{
    Executable code;
}
```

### Example:

```
<?php
    function func()
    {
```

```
        echo "This is PHP program using Functions";  
    }  
    func();  
?>
```

**Output:**

This is PHP program using Functions

**FUNCTION PARAMETERS OR ARGUMENTS**

- ❖ The information or variable, within the function's parenthesis, are called parameters.
- ❖ These are used to hold the values executable during runtime.
- ❖ A user is free to take in as many parameters as he wants, separated with a comma(,) operator. These parameters are used to accept inputs during runtime.
- ❖ While passing the values like during a function call, they are called arguments.
- ❖ An argument is a value passed to a function and a parameter is used to hold those arguments. In common term, both parameter and argument mean the same.

**Syntax:**

```
function function_name($first_parameter, $second_parameter)  
{  
    executable code;  
}
```

**Example:**

```
<?php  
    function pro($num1, $num2, $num3)
```

```
{  
    $product = $num1 * $num2 * $num3;  
    echo "The product is $product";  
}  
pro(2, 3, 5);  
?>
```

**Output:** The product is 30

### SETTING DEFAULT VALUES FOR FUNCTION PARAMETER

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

#### Example:

```
<?php  
function defk($str, $num=12)  
{  
    echo "$str is $num years old \n";  
}  
defk("Ram", 15);  
defk("Adam");  
?>
```

**Output:** Ram is 15 years old

Adam is 12 years old

In the above example, the parameter \$num has a default value 12, if we do not pass any value for this parameter in a function call then this default value 12 will be considered. Also the parameter \$str has no default value, so it is compulsory.

### RETURNING VALUES FROM FUNCTIONS

- ❖ Functions can also return values to the part of program from where it is called.
- ❖ The return keyword is used to return value back to the part of program, from where it was called.
- ❖ The returning value may be of any type including the arrays and objects.
- ❖ The return statement also marks the end of the function and stops the execution after that and returns the value.

#### Example:

```
<?php
    function pro($num1, $num2, $num3)
    {
        $product = $num1 * $num2 * $num3;
        return $product;
    }
    $retValue = pro(2, 3, 5);
    echo "The product is $retValue";
?>
```

**Output:** The product is 30

### PARAMETER PASSING TO FUNCTIONS

- ❖ PHP allows us two ways in which an argument can be passed into a function:
- 1. Pass by Value:** On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged. That means a duplicate of the original value is passed as an argument.

**2. Pass by Reference:** On passing arguments as pass by reference, the original value is passed. Therefore, the original value gets altered. In pass by reference we actually pass the address of the value, where it is stored using ampersand sign(&).

**Example:**

```
<?php
    function val($num)
    {
        $num += 2;
        return $num;
    }
    function ref (&$num)
    {
        $num += 10;
        return $num;
    }
    $n = 10;
    val($n);
    echo "The original value is still $n \n";
    ref($n);
    echo "The original value changes to $n";
?>
```

**Output:**    The original value is still 10  
                 The original value changes to 20

### VARIABLE FUNCTIONS

❖ PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a

function with the same name as whatever the variable evaluates to, and will attempt to execute it.

- ❖ Among other things, this can be used to implement callbacks, function tables, and so forth.

**Example:**

```
function foo()
{
    echo "hi<br />";
}

function bar($arg = "")
{
    echo " argument was '$arg'.<br />n";
}

function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();      // This calls foo()
$func = 'bar';
$func('test'); // This calls bar()
$func = 'echoit';
$func('test'); // This calls echoit()
?>
```

**Output:** hi

argument was 'test'.

test

### RECURSIVE FUNCTIONS

- ❖ A recursive function is a function that calls itself again and again until a condition is satisfied.
- ❖ Recursive functions are often used to solve complex mathematical calculations, or to process deeply nested structures e.g., printing all the elements of a deeply nested array.

#### Example:

```
<?php
    function factorial($n)
    {
        if ($n==0)
        {
            return 1;
        }
        else
        {
            return $n * factorial($n - 1);
        }
    }
    echo factorial(4), "\n";
    echo factorial(10), "\n";
?>
```

**Output:** 24  
3628800

**Write a php program to calculate sum and average of array elements.**

```
<?php
$scores = array(1,4,6,4,3,5,6,5,3,5,6);
$score_count = count($scores);
$score_sum = array_sum($scores);
echo "the sum is" .$score_sum."<br>";
$mean_average = $score_sum / $score_count;
echo "<p>The Mean Average of the scores is
$mean_average</p>";
?>
```

**Output:** the sum is42

The Average of the scores is 4.2

**Write a function to calculate sum and average of array elements.**

```
<?php
function avg( $a, $b)
{
    $avg=$b/$a;
    return $avg;
}

function sum( $arr, $n)
{
    $sum = 0;
    for ($i = 0; $i < $n; $i++)
        $sum += $arr[$i];
    return $sum;
}
```



```
$arr =array(12, 3, 4, 15);  
$n = sizeof($arr);  
$sum1=sum($arr, $n);  
echo "Sum of given array is ".$sum1."<br>";  
$avg1=avg($n,$sum1);  
echo "Avg of given array is ".$avg1;  
?>
```

**Output:** Sum of given array is 34

Avg of given array is 8.5

**How to search the element in an array with suitable program**

```
<?php  
$array = array(0 , 1 , 2 , 3 );  
$key=array_search(2, $array)?"Element exist":"element not exist";  
echo $key;  
?>
```

**Output:** Element exist

**Write a function in PHP to take 'n' as argument and return the factorial of a given number.**

```
<?php  
function Factorial($number)  
{  
    $factorial = 1;  
    for ($i = 1; $i <= $number; $i++)  
    {  
        $factorial = $factorial * $i;  
    }  
    return $factorial;  
}
```

```
}  
$n = 5;  
$fact = Factorial($n);  
echo "Factorial = $fact";  
?>
```

**Output:**     Factorial = 120

# PHP PROGRAMMING

---

## UNIT III

### Introduction to Strings:

Creating and accessing Strings, String Related Library functions.

### File handling in Php:

Defining a File, different file operations.

**STRINGS:** String is a sequence of characters. For example 'rgmcet' or "rgmcet" is string. Everything inside the quotes single(' ') or double(" ") in php can be treated as string. There are 2 ways to specify string in PHP

**1. single quoted strings:** We can create a string by enclosing text in a single quote. This type of strings does not process special characters inside quotes.

#### Example

```
<?php
    $str='rgmcet';
    echo ' welcome to $str ';
?>
```

**Output:** welcome to \$str

In the above program the echo statement prints the variable name rather than printing the contents of the variables. This is because single quote strings in php does not process special characters. Hence the string is unable to identify the \$ sign as start of a variable name.

**2. double quoted strings:** we can specify string by enclosing text within double quotes. This type of strings can process special characters inside quotes.

#### Example

```
<?php
    echo "Hello php I am double quote String\n";
    $str="rgmcet";
    echo " welcome to $str";
?>
```

**Output:** Hello php I am double quote String

# PHP PROGRAMMING

---

welcome to rgmcet

In the above program we can see that the double quote string is processing the special characters according to their properties. The \n character is not printed and is considered as a newline. Also instead of the variable name rgmcet is printed

## Example

```
<?php
    $str="Hello "php" I am double quote String";
    echo $str;

?>
```

**Output:** Parse error: syntax error, unexpected 'quote' (T\_STRING) in C:\wamp\www\string1.php on line 2

## Example

```
<?php
    $str="Hello \"php\" I am double quote String";
    echo $str;

?>
```

**Output:** Hello php I am double quote String

**Note:** Using double quote String you can also display variable value on webpage

## CONCATENATION OF TWO STRINGS

There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side.

## Example1:

```
<?php
    $a = 'Hello';
    $b = 'World!';
    $c = $a.$b;
```

```
echo " $c \n";
```

```
?>
```

**Output:** HelloWorld!

### Example2:

```
<?php
```

```
    $fname = 'John';
```

```
    $lname = 'Carter!';
```

```
    $c = $fname." ".$lname;
```

```
    echo " $c \n";
```

```
?>
```

**Output:** John Carter!

### Example3:

```
<?php
```

```
$a = 'Hello';
```

```
$a. = "World!";
```

```
echo " $a \n";
```

```
?>
```

**Output:** HelloWorld!

## STRING FUNCTIONS IN PHP

PHP have lots of predefined function which is used to perform operation with string some functions are:

**strlen():** strlen() function returns the length of a string.

### Example

```
<?php
```

```
    echo strlen("Hello world!");
```

```
?>
```

**Output:** 12

**str\_word\_count():** str\_word\_count() function is used to count numbers of words in given string.

### Example

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

**Output:** 2

**strrev():** strrev() function is used to reverse any string.

**Example**

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

**Output:** !dlrow olleH

**strtolower():** strtolower() function is used to convert uppercase letter into lowercase letter.

**Example**

```
<?php
    $str="Hello friends i am HITESH";
    $str=strtolower($str);
    echo $str;
?>
```

**Output:** hello friends i am hitesh

**strtoupper():** PHP strtoupper() function is used to convert lowercase letter into uppercase letter.

**Example**

```
<?php
    $str="Hello friends i AM Hitesh";
    $str=strtoupper($str);
    echo $str;
?>
```

**Output:** HELLO FRIENDS I AM HITESH

**ucwords():** ucwords() function is used to convert first letter of every word into upper case.

## Example

```
<?php
    $str="hello friends i am hitesh";
    $str=ucwords($str);
    echo $str;

?>
```

**Output:** Hello Friends I Am Hitesh

**ucfirst():** ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

## Example

```
<?php
    function firstUpper($string)
    {
        return(ucfirst($string));
    }
    $string="welcome to GeeksforGeeks";
    echo (firstUpper($string));

?>
```

**Output:** Welcome to GeeksforGeeks

**lcfirst():** lcfirst() function is used to convert first character into lowercase. It doesn't change the case of other characters.

## Example

```
<?php
    function firstLower($string)
    {
        return(lcfirst($string));
    }
    $string="WELCOME to GeeksforGeeks";
    echo (firstLower($string));

?>
```

## PHP PROGRAMMING

---

**Output:** wELCOME to GeeksforGeeks

**strstr()**(or) **strchr()**: The strstr() function searches for the first occurrence of a string inside another string.

**Syntax:** strstr(string,search)

**Example:**

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

**output:** world!

**Example:** Search a string for the ASCII value of "o" and return the rest of the string:

```
<?php
echo strstr("Hello world!",111);
?>
```

**Output:** o world!

**Example:**

```
<?php
$originalStr = "geeks for geeks";
$searchStr = "geeks" ;
echo strstr($originalStr, $searchStr);
?>
```

**Output:** geeks for geeks

**Program 2:** Program to demonstrate strstr() function when word is not found.

```
<?php
$originalStr = "geeks for geeks";
$searchStr = "gfg" ;
echo strstr($originalStr, $searchStr);
?>
```

**Output:** -NIL-

**strcmp()**: Compare two strings with case-sensitive manner



## PHP PROGRAMMING

---

**Syntax:**     strcmp(string1,string2)

**This function returns:**

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

**Example:**

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

**Output:**     0  
              -1

**Example:**

```
<?php
echo strcmp("Hello world!","Hello world!")."<br>";
echo strcmp("Hello world!","Hello")."<br>"; // string1 is greater than string2
echo strcmp("Hello world!","Hello world! Hello!")."<br>"; // string1 is less than
string2
?>
```

**Output:**     0  
              7  
              -7

**strcasecmp():** Compare two strings with case-insensitive manner

**Syntax:**     strcasecmp(string1,string2)

**This function returns:**

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

**Example:**

```
<?php
echo strcasecmp("Hello","HELLO");
echo strcasecmp("Hello","hELLo");
?>
```

**Output:**     0  
              0

**Example:**

```
<?php
echo strcasecmp("Hello world!","HELLO WORLD!"); // The two strings are equal
echo strcasecmp("Hello world!","HELLO"); // String1 is greater than string2
echo strcasecmp("Hello world!","HELLO WORLD! HELLO!"); // String1 is less
than string2
?>
```

**Output:**     0  
              7  
             -7

**strncmp():** The strncmp() is used to compare first n character of two strings. This function is case-sensitive which points that capital and small cases will be treated differently, during comparison.

**Syntax:**     strncmp( \$str1, \$str2, \$len )

This function returns:

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

**Example:**

```
<?php
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for Geeks ";
// Both the strings are equal
$test=strncmp($str1, $str2, 16 );
```

```
echo "$test";
```

```
?>
```

**Output:** 0

**Example:**

```
?php
```

```
// Input strings
```

```
$str1 = "Geeks for Geeks ";
```

```
$str2 = "Geeks for ";
```

```
$test=strncmp($str1, $str2, 16 );
```

```
// In this case the second string is smaller
```

```
echo "$test";
```

```
?>
```

**Output:** 6

**Example:**

```
<?php
```

```
// Input Strings
```

```
$str1 = "Geeks for ";
```

```
$str2 = "Geeks for Geeks ";
```

```
$test=strncmp($str1, $str2, 16 );
```

```
// In this case the first string is smaller
```

```
echo "$test";
```

```
?>
```

**Output:** -6

**strncasecmp()**: The `strncmp()` is used to compare first n character of two strings. This function is case-sensitive which points that capital and small cases will be treated differently, during comparison.

**Syntax:** `strncasecmp( $str1, $str2, $len )`

This function returns:

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

**Example:**

```
<?php
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for Geeks ";
// Both the strings are equal
$test=strncasecmp($str1, $str2, 16 );
echo "$test";
?>
```

**Output :** 0

**Example:**

```
?php
// Input strings
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for ";
$test=strncasecmp($str1, $str2, 16 );
// In this case the second string is smaller
echo "$test";
?>
```

**Output:** 6

**Example:**

```
<?php
// Input Strings
$str1 = "Geeks for ";
$str2 = "Geeks for Geeks ";
$test=strncasecmp($str1, $str2, 16 );
// In this case the first string is smaller
echo "$test";
?>
```

**Output:** -6

**strpos()** It enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return "False".

**Syntax:** Strpos(string,text);

**Example**

```
<?php
echo strpos("Welcome to Cloudways","Cloudways");
?>
```

**Output:** 11

**Str\_replace():** It used for replacing specific text within a string.

**Syntax:** Str\_replace(string to be replaced,text,string)

**Example**

```
<?php
echo str_replace("cloudways", "the programming world", "Welcome to
cloudways");
?>
```

**Output:** Welcome to the programming world

**str\_repeat():** This function is used for repeating a string a specific number of times.

**Syntax:** Str\_repeat(string,repeat)

**Example**

```
<?php
echo str_repeat("=",13);
?>
```

**Output:** =====

**substr():** This function you can display or extract a string from a particular position.

**Syntax:** *substr(string,start,length)*

**Example**

```
<?php
```

## PHP PROGRAMMING

---

```
echo substr("Welcome to Cloudways",6)."<br>";
echo substr("Welcome to Cloudways",0,10)."<br>";
?>
```

**Output:**      e to Cloudways  
                 Welcome to

**chunk\_split():**The chunk\_split() function splits a string into a series of smaller parts.

**Syntax:**      chunk\_split(string,length,end)

**Example:** Split the string after each sixth character and add a "..." after each split:

```
<?php
$str = "Hello world!";
echo chunk_split($str,6,"...");
?>
```

**Output:**      Hello ...world!...

**trim():**Removes whitespace or other characters from both sides of a string

Parameter	Description
string	Required. Specifies the string to check
charlist	Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed: "\0" - NULL "\t" - tab "\n" - new line "\x0B" - vertical tab "\r" - carriage return " " - ordinary white space

**Example:**

```
<?php
echo trim(" testing ")."<br>";
echo trim(" testing ", " teng");
```

```
?>
```

**Output:**     testing  
             sti

**ltrim()**: Removes whitespace or other characters from the left side of a string

**Syntax:**     ltrim(string,charlist)

```
<?php
```

```
$str = "Hello World!";  
echo $str . "<br>";  
echo ltrim($str,"Hello");
```

```
?>
```

**Output:**     Hello World!  
             World!

**rtrim()**: Removes whitespace or other characters from the right side of a string

**Syntax:**     rtrim(string,charlist)

```
<?php
```

```
$str = "Hello World!";  
echo $str . "<br>";  
echo rtrim($str,"World!");
```

```
?>
```

**Output:**     Hello World!  
             Hello

**Write a PHP program to count number of characters in a given string.**

```
<?php
```

```
$text="w3resource";  
$search_char="r";  
$count="0";  
for($x="0"; $x< strlen($text); $x++)  
{  
    $count=$count+1;  
}
```

## PHP PROGRAMMING

---

```
echo $count."\n";
```

```
?>
```

### Function

### Description

chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
count_chars()	Returns information about characters used in a string
echo()	Outputs one or more strings
explode()	Breaks a string into an array
fprintf()	Writes a formatted string to a specified output stream
parse_str()	Parses a query string into variables
print()	Outputs one or more strings
printf()	Outputs a formatted string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_getcsv()	Parses a CSV string into an array
str_ireplace()	Replaces some characters in a string (case-insensitive)
str_pad()	Pads a string to a new length
str_replace()	Replaces some characters in a string (case-sensitive)
str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
stristr()	Finds the first occurrence of a string inside another string (case-insensitive)
strrchr()	Finds the last occurrence of a string inside another string
strripos()	Finds the position of the last occurrence of a string



## PHP PROGRAMMING

---

	inside another string (case-insensitive)
strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
substr()	Returns a part of a string
substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
substr_count()	Counts the number of times a substring occurs in a string
substr_replace()	Replaces a part of a string with another string

### FILE HANDLING

- ❖ A file is simply a resource for storing information on a computer.
- ❖ Files are usually used to store information such as;
  1. Configuration settings of a program
  2. Simple data such as contact names against the phone numbers.
  3. Images, Pictures, Photos, etc.

### PHP FILE FORMATS SUPPORT

- ❖ PHP file functions support a wide range of file formats that include;
  1. File.txt
  2. File.log
  3. File.custom\_extension i.e. file.xyz
  4. File.csv
  5. File.gif, file.jpg etc
  6. Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.
  7. You want to store simple data such as server logs for later retrieval and analysis

8. You want to store program settings i.e. program.ini

### FILE ATTRIBUTES

File attributes are the properties of a file, for example its size, the last time it was accessed, its owner, etc.

#### filesize()

- ❖ The function filesize() retrieves the size of the file in bytes.

#### Example:

```
<?php
    $f = "C:\Windows\win.ini";
    $size = filesize($f);
    echo $f . " is " . $size . " bytes.";
?>
```

When executed, the example code displays:

C:\Windows\win.ini is 510 bytes.

If the file doesn't exist, the filesize() function will return false and emit an E\_WARNING.

#### file\_exists()

- ❖ This function is used to check first whether the file exists or not.
- ❖ It comes in handy when we want to know if a file exists or not before processing it.
- ❖ You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.
- ❖ Syntax:

```
<?php
    file_exists($filename);
?>
```

Here,

“file\_exists()” is the PHP function that returns true if the file exists and false if it does not exist.

“\$file\_name” is the path and name of the file to be checked

**Example:**

```
<?php
    $f = "C:\Windows\win.ini";
    if (file_exists($f))
    {
        $size = filesize($f);
        echo $f . " is " . $size . " bytes.";
    }
    else
    {
        echo $f . " does not exist.";
    }
}
```

**FILE HISTORY**

To determine when a file was last accessed, modified, or changed, you can use the following functions respectively: `fileatime()`, `filemtime()`, and `filectime()`.

**Example:**

```
<?php
    $dateFormat = "D d M Y g:i A";
    $atime = fileatime($f);
    $mtime = filemtime($f);
    $ctime = filectime($f);
    echo $f . " was accessed on " . date($dateFormat, $atime) . "<br>";
    echo $f . " was modified on " . date($dateFormat, $mtime) . "<br>";
    echo $f . " was changed on " . date($dateFormat, $ctime) . ".";
?>
```

The code here retrieves the timestamp of the last access, modifies, and change dates and displays them,

C:\Windows\win.ini was accessed on Tue 14 Jul 2009 4:34 AM.

C:\Windows\win.ini was modified on Fri 08 Jul 2011 2:03 PM.

C:Windows\win.ini was changed on Tue 14 Jul 2009 4:34 AM.

To clarify, `filemtime()` returns the time when the contents of the file was last modified, and `filectime()` returns the time when information associated with the file, such as access permissions or file ownership, was changed.

### FILE PERMISSIONS

Before working with a file you may want to check whether it is readable or writeable to the process. For this you'll use the functions **`is_readable()`** and **`is_writable()`**

#### Example:

```
<?php
    echo $f . (is_readable($f) ? " is" : " is not") . " readable.<br>";
    echo $f . (is_writable($f) ? " is" : " is not") . " writeable.";
?>
```

Both functions return a Boolean value whether the operation can be performed on the file. Using the ternary operator you can tailor the display to state whether the file is or is not accessible as appropriate.

C:Windows\win.ini is readable.

C:Windows\win.ini is not writeable.

### FILE OR NOT?

To make absolutely sure that you're dealing with a file you can use the `is_file()` function. `is_dir()` is the counterpart to check if it is a directory.

#### Example:

```
<?php
    echo $f . (is_file($f) ? " is" : " is not") . " a file.<br>";
    echo $f . (is_dir($f) ? " is" : " is not") . " a directory.";
?>
```

output:

C:Windows\win.ini is a file.

C:Windows\win.ini is not a directory.

## OPENING AND CLOSING FILES

**fopen()**: The PHP `fopen()` function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

**Syntax:**     **`fopen($file_name,$mode);`**

Here,

“`fopen`” is the PHP open file function

“`$file_name`” is the name of the file to be opened

“`$mode`” is the mode in which the file should be opened

❖ Files modes can be specified as one of the six options in this table.

**r**

Opens the file for reading only.

Places the file pointer at the beginning of the file.

**r+**

Opens the file for reading and writing.

Places the file pointer at the beginning of the file.

**w**

Opens the file for writing only.

Places the file pointer at the beginning of the file and truncates the file to zero length. If files does not exist then it attempts to create a file.

**w+**

Opens the file for reading and writing only.

Places the file pointer at the beginning of the file and truncates the file to zero length. If files does not exist then it attempts to create a file.

**a**

Opens the file for writing only.

Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

**a+**

Opens the file for reading and writing only.

Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

- ❖ If an attempt to open a file fails then fopen returns a value of false otherwise it returns a file pointer which is used for further reading or writing to that file.

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
    echo ( "Error in opening file" );
    exit();
}
?>
```

**fclose():** After making a changes to the opened file it is important to close it with the fclose() function. The fclose() function requires a file pointer as its argument and then returns true when the closure succeeds or false if it fails.

**Syntax:    fclose(\$handle);**

Here,

“fclose” is the PHP function for closing an open file

“\$handle” is the file pointer resource.

### READING A FILE

**fread():**

- ❖ Once a file is opened using fopen() function it can be read with a function called fread().
- ❖ This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.
- ❖ The files length can be found using the filesize() function which takes the file name as its argument and returns the size of the file expressed in bytes.
- ❖ The steps required to read a file with PHP.

# PHP PROGRAMMING

---

1. Open a file using fopen() function.
2. Get the file's length using filesize() function.
3. Read the file's content using fread() function.
4. Close the file with fclose() function.

## Example

```
<?php
    $filename = "tmp.txt";
    $file = fopen( $filename, "r" );
    if( $file == false )
    {
        echo ( "Error in opening file" );
        exit();
    }
    $filesize = filesize( $filename );
    $filetext = fread( $file, $filesize );
    fclose( $file );
    echo "File size : $filesize bytes" ;
    echo "$filetext" ;

?>
```

## Output

File size : 278 bytes

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications. This tutorial helps you to build your base with PHP.

## file\_get\_contents()

- ❖ This function will read the entire contents of a file into a variable without the need to open or close the file.

## Example:

```
<?php
```

```
$f = "c:\windows\win.ini";  
$f1 = file_get_contents($f);  
echo $f1;
```

?>

### Reading a file Line by Line

**fgets():** The fgets() function is used to read a single line from a file, and after a call to this function, the file pointer has moved to the next line.

Syntax:      fgets(filepointer);

### Example:

```
<?php
```

```
    $f = "c:\windows\win.ini";  
    $f1 = fopen($f, "r");  
    do  
    {  
        echo fgets($f1) . "<br>";  
    }while (!feof($f1));  
    fclose($f1);
```

?>

### Example:

```
<?php
```

```
$fileName = "/doc/myFile.txt";  
$fp = fopen($fileName,"r");  
if( $fp == false )  
{  
    echo ( "Error in opening file" );  
    exit();  
}  
while(!feof($fp))  
{  
    echo fgets($fp). "<br>";
```



```
}  
?>
```

### **fgetc()**

#### **Example:**

```
<?php  
$file = fopen("test2.txt","r");  
while (! feof ($file))  
{  
    echo fgetc($file);  
}  
fclose($file);  
?>
```

**output:** Hello, this is a test file.

**feof():** The function feof() checks whether the filepointer has reached its end of the file or not.

#### **Example:**

```
<?php  
$f = "c:\windows\win.ini";  
$f1 = fopen($f, "r");  
do  
{  
    echo fgets($f1) . "<br>";  
}while (!feof($f1));  
fclose($f1);  
?>
```

## **WRITING A FILE**

### **fwrite()**

The fwrite() function in PHP is an inbuilt function which is used to write to an open file. The fwrite() function stops at the end of the file or when it reaches the specified length passed as a parameter, whichever comes first.

# PHP PROGRAMMING

---

Syntax:      fwrite(filepointer, \$string, \$length);

Here,

“fwrite” is the PHP function for writing to files

“\$filepointer” is the file pointer resource

“\$string” is the data to be written in the file.

“\$length” is optional, can be used to specify the maximum file length.

## Example

```
<?php
```

```
    $filename = "/home/user/guest/newfile.txt";
```

```
    $file = fopen( $filename, "w" );
```

```
    if( $file == false )
```

```
    {
```

```
        echo ( "Error in opening new file" );
```

```
        exit();
```

```
    }
```

```
    fwrite( $file, "This is a simple test\n" );
```

```
    fclose( $file );
```

```
?>
```

```
<html>
```

```
    <head>
```

```
        <title>Writing a file using PHP</title>
```

```
    </head>
```

```
    <body>
```

```
        <?php
```

```
            $filename = "newfile.txt";
```

```
            $file = fopen( $filename, "r" );
```

```
            if( $file == false )
```

```
            {
```

```
                echo ( "Error in opening file" );
```

```
                exit();
```

```
}
$filesize = filesize( $filename );
$filetext = fread( $file, $filesize );
fclose( $file );
echo ( "File size : $filesize bytes" );
echo ( "$filetext" );
echo("file name: $filename");

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

### Output

```
File size : 23 bytes
This is a simple test
file name: newfile.txt
```

### file\_put\_contents()

- ❖ This function is used to write the new contents into a file.
- ❖ First, it deletes the existing contents of a file then it writes new contents.
- ❖ If file is not available it creates a new file.

### Example

```
<?php
    file_put_contents("abc.txt","Uday");

?>

<?php
$myfile = fopen("gfg.txt", "w");
echo fputs($myfile, "Geeksforgeeks is a portal of geeks!", 13);
fclose($myfile);
fopen("gfg.txt", "r");
echo fread($myfile, filesize("gfg.txt"));
fclose($myfile);
```

?>

**fputs( ):** The fputs() function in PHP is an inbuilt function which is used to write to an open file. The fputs() function stops at the end of the file or when it reaches the specified length passed as a parameter, whichever comes first.

**Syntax:** fputs(filepointer, string, length)

The fputs() function in PHP accepts three parameters.

**file:** It is a mandatory parameter which specifies the file.

**string :** It is a mandatory parameter which specifies the string to be written.

**length :** It is an optional parameter which specifies the maximum number of bytes to be written.

**Example:**

```
<?php
$file = fopen("test.txt","w");
echo fputs($file,"Hello World. Testing!");
fclose($file);
?>
```

**output:** 21

**Example:**

```
<?php
$myfile = fopen("gfg.txt", "w");
echo fputs($myfile, "Geeksforgeeks is a portal of geeks!", 13);
fclose($myfile);
fopen("gfg.txt", "r");
echo fread($myfile, filesize("gfg.txt"));
fclose($myfile);
?>
```

**Output:** Geeksforgeeks

**ftell( ):** The ftell() function is used to return the current position in an open file. It returns the current file pointer position on success, or FALSE on failure.

**Syntax:**      `ftell( $filepointer )`

**Example:**

```
<?php
$myfile = fopen("gfg.txt", "r");
echo ftell($myfile);
fclose($myfile);
?>
```

**fseek():** This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes. This function returns 0 on success, or -1 on failure

**Syntax:**    `fseek(filepointer,offset,whence)`

Parameter	Description
file	Required. Specifies the open file to seek in
offset Required.	Specifies the new position (measured in bytes from the beginning of the file)
whence	Optional. SEEK_SET - Set position equal to offset. Default SEEK_CUR - Set position to current location plus offset SEEK_END - Set position to EOF plus offset

**Example:**

```
<?php
$myfile = fopen("gfg.txt", "r");
echo ftell($myfile);
fseek($myfile, "36");
echo "<br />" . ftell($myfile);
fclose($myfile);
?>
```

**rewind( ):** The rewind() is used to set the position of the file pointer to the beginning of the file.

**Syntax:**        rewind(filepointer)

```
<?php
$myfile = fopen("gfg.txt", "r+");
fwrite($myfile, 'geeksforgeeks a computer science portal');
rewind($myfile);
fwrite($myfile, 'geeksportal');
rewind($myfile);
echo fread($myfile, filesize("gfg.txt"));
fclose($myfile);

?>
```

### COPYING A FILE

❖ The PHP copy function is used to copy files.

❖ Syntax:

```
<?php
    copy($file,$copied_file);

?>
```

Here,

“\$file” specifies the file path and name of the file to be copied.

“copied\_file” specified the path and name of the copied file

### Example

```
<?php
    copy('my_settings.txt', 'my_settings_backup.txt') or die("Could not copy
    file");
    echo "File successfully copied to 'my_settings_backup.txt'";

?>
```

### DELETING A FILE

❖ The unlink function is used to delete the file.

### Example

```
<?php
    if (!unlink('my_settings_backup.txt'))
    {
        echo "Could not delete file";
    }
    else
    {
        echo "File 'my_settings_backup.txt' successfully deleted";
    }
?>
```

### RENAMING FILES

- ❖ This function is used to rename the existing file name with the specified file name.

### Example

```
<?php
    $file = "file.txt";
    if(file_exists($file))
    {
        if(rename($file, "newfile.txt"))
        {
            echo "File renamed successfully.";
        }
        else
        {
            echo "ERROR: File cannot be renamed.";
        }
    }
    else
    {
```

```
        echo "ERROR: File does not exist.";
    }
?>
```

### FILE SYSTEM FUNCTIONS

Function	Description
filetype()	Returns the type of the file.
is_dir()	Checks whether the file is a directory.
is_executable()	Checks whether the file is executable.
rmdir()	Removes an empty directory.
readfile()	Read the contents of file and write those contents in current buffer.

### DIRECTORY FUNCTIONS

mkdir	Creates a new directory.
rmdir	Removes a directory.
opendir	Opens a directory.
readdir	Reads the contents of directory.
closedir	Closes the opened directory.
scandir	Gets all files of a directory as array.
getcwd	Get current working directory.
chdir	Changes the current working directory.



# PHP PROGRAMMING

---

## UNIT IV

### INTRODUCTION TO OOPS: INTRODUCTION OBJECTS, DECLARING A CLASS, PROPERTIES AND METHODS, INHERITANCE, POLYMORPHISM & ENCAPSULATION, CONSTRUCTOR, DESTRUCTOR, EXTENDING CLASSES, USING \$THIS, USING ACCESS SPECIFIERS, ABSTRACT METHOD AND CLASS, USING INTERFACE.

Object-oriented programming (OOP) is a programming language model in which programs are organized around data, or [objects](#), rather than functions and logic.

OOP (not Oops!) refers to a programming methodology based on objects, instead of just functions and procedures(logic).

The three major principles of OOP are;

**Encapsulation:** Wrapping some data in single unit is called Encapsulation. Encapsulation is used to safe data or information in an object from other it means encapsulation is mainly used for protection purpose (or) **Encapsulation** is a concept of wrapping up or binding up related data members and methods in a single module known as encapsulation

**Inheritance:**

## PHP PROGRAMMING

---

**Polymorphism:** **polymorphism** means the ability to have many forms. In other words, "**Polymorphism** describes a pattern in Object Oriented Programming in which a class has varying functionality while sharing a common interfaces."

**DECLARING A CLASS:** Classes are the blueprints of objects(or) Class is a collection of objects(or) Wrapping up of the data and associated methods into a single unit is called class. A class can be declared using the `class` keyword, followed by the name of the class and a pair of curly braces (`{}`).

```
<?php
```

```
    class person
```

```
    {
```

```
    }
```

```
?>
```

**Example:**

```
<?php
```

```
class GeeksforGeeks
```

```
{
```

```
    public function __construct()
```

```
    {
```

```
        echo 'The class "' . __CLASS__ . '" was initiated!<br>';
```

```
    }
```

```
}
```

```
$obj = new GeeksforGeeks;
```

```
?>
```

**Output:**        The class "GeeksforGeeks" was initiated.

```
<?php
```

```
    class Mobile
```

```
    {
```

---

**Prepared By: Dept Of CSE, RGM CET        Page 2**

Creating an object from an existing class is called instantiating the objec

## PHP PROGRAMMING

---

```
var $price;

var $title;

    function setPrice($par)

    {

        $this->price = $par;

    }

function getPrice(){

    echo $this->price ."";

}

}

$Samsung = new Mobile();

$Samsung->setPrice( 90000 );

$Samsung->getPrice();

?>
```

Object: It is an Instance of a class (or) It is a

### **Creating an Object:**

Following is an example of how to create object using new operator.

```
class Book {

    // Members of class Book

}

// Creating three objects of book
```

---

**Prepared By: Dept Of CSE, RGM CET      Page 3**

Creating an object from an existing class is called instantiating the objec

## PHP PROGRAMMING

---

```
$physics = new Books;
```

```
$maths = new Books;
```

```
$chemistry = new Books;
```

### Member Functions:

After creating our objects, we can call member functions related to that object. A member function typically accesses members of current object only.

Example:

```
$physics->setTitle( "Physics for High School" );
```

```
$chemistry->setTitle( "Advanced Chemistry" );
```

```
$maths->setTitle( "Algebra" );
```

```
$physics->setPrice( 10 );
```

```
$chemistry->setPrice( 15 );
```

```
$maths->setPrice( 7 );
```

### Example:

```
<?php
```

```
class Books {  
    /* Member variables */  
    var $price;  
    var $title;  
    /* Member functions */  
    function setPrice($par){
```

## PHP PROGRAMMING

---

```
        $this->price = $par;

    }

    function getPrice(){

        echo $this->price."<br>";

    }

    function setTitle($par){

        $this->title = $par;

    }

    function getTitle(){

        echo $this->title."<br>";

    }

}

/* Creating New object using "new" operator */

$maths = new Books;

/* Setting title and prices for the object */

$maths->setTitle( "Algebra" );

$maths->setPrice( 7 );

/* Calling Member Functions */

$maths->getTitle();
```

# PHP PROGRAMMING

---

```
$maths->getPrice();
```

```
?>
```

## Defining Class Properties

To add data to a class, properties, or class-specific variables, are used. These work exactly like regular variables, except they're bound to the object and therefore can only be accessed using the object.

```
<?php

class MyClass
{
    public $prop1 = "I'm a class property!";
}

$obj = new MyClass;

echo $obj->prop1; // Output the property

?>
```

**Output:** I'm a class property!

## Defining Class Methods

Methods are class-specific functions. Individual actions that an object will be able to perform are defined within the class as methods.

```
<?php

class MyClass
{
    public $prop1 = "I'm a class property!";

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }
}
```

## PHP PROGRAMMING

---

```
}

public function getProperty()

{

    return $this->prop1 . "<br />";

}

}

$obj = new MyClass;

echo $obj->getProperty(); // Get the property value

$obj->setProperty("I'm a new property value!"); // Set a new one

echo $obj->getProperty(); // Read it out again to show the change

?>
```

Output:            I'm a class property!

                  I'm a new property value!

### Example:

```
<?php

class MyClass

{

    public $prop1 = "I'm a class property!";

    public function setProperty($newval)

    {

        $this->prop1 = $newval;
```

## PHP PROGRAMMING

---

```
}

public function getProperty()

{

    return $this->prop1 . "<br />";

}

}

// Create two objects

$obj = new MyClass;

$obj2 = new MyClass;

// Get the value of $prop1 from both objects

echo $obj->getProperty();

echo $obj2->getProperty();

// Set new values for both objects

$obj->setProperty("I'm a new property value!");

$obj2->setProperty("I belong to the second instance!");

// Output both objects' $prop1 value

echo $obj->getProperty();

echo $obj2->getProperty();

?>
```

**Output:**        I'm a class property!

                 I'm a class property!



# PHP PROGRAMMING

---

I'm a new property value!

I belong to the second instance!

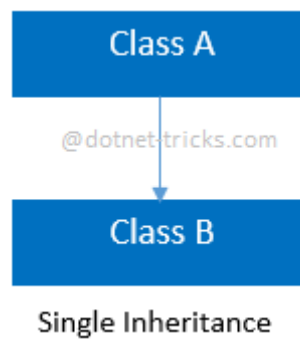
## Different Types of Inheritance

OOPs support the six different types of inheritance as given below :

### Single inheritance

In this inheritance, a derived class is created from a single base class. In the given example, Class A is the parent class and Class B is the child class since Class B inherits the features and behavior of the parent class A.

1.



### Syntax for Single Inheritance

```
//Base Class
class A
{
    public void fooA()
    {
        //TO DO:
    }
}

//Derived Class
class B : A
{
}
```

## PHP PROGRAMMING

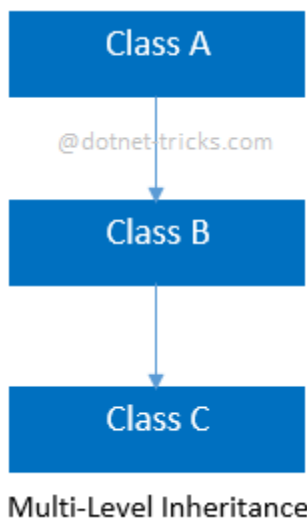
---

```
public void fooB()
{
//TO DO:
}
```

### 2. Multi-level inheritance

In this inheritance, a derived class is created from another derived class.

In the given example, class c inherits the properties and behavior of class B and class B inherits the properties and behavior of class A. So, here A is the parent class of B and class B is the parent class of C. So, here class C implicitly inherits the properties and behavior of class A along with Class B i.e there is a multilevel of inheritance.



#### Syntax for Multi-level Inheritance

```
//Base Class
class A
{
public void fooA()
{
//TO DO:
}
```

## PHP PROGRAMMING

---

```
}  
  
//Derived Class  
class B : A  
{  
    public void fooB()  
    {  
        //TO DO:  
    }  
}
```

```
//Derived Class  
class C : B  
{  
    public void fooC()  
    {  
        //TO DO:  
    }  
}
```

### 3. Multiple inheritance

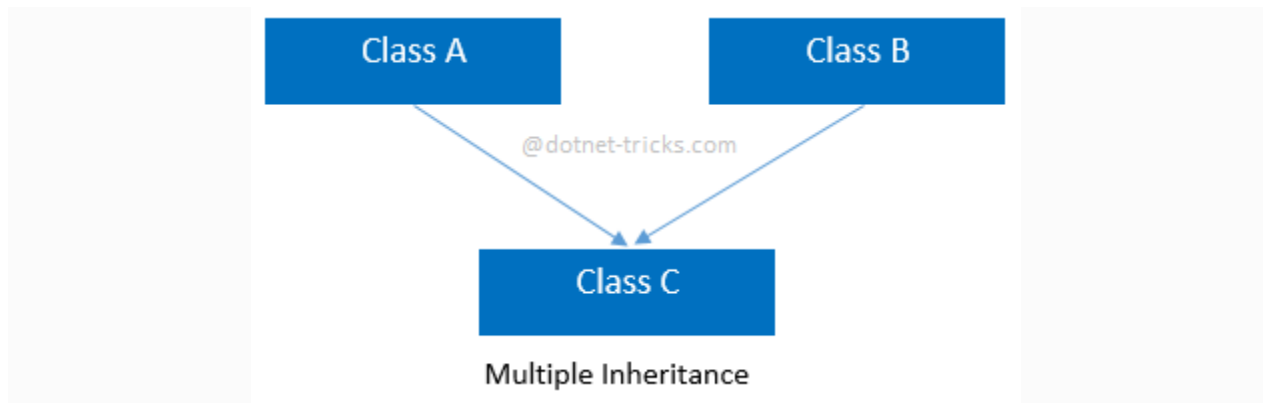
#### Multiple inheritance

In this inheritance, a derived class is created from more than one base class. This inheritance is not supported by [.NET](#) Languages like C#, F# etc. and Java Language.

In the given example, class c inherits the properties and behavior of class B and class A at same level. So, here A and Class B both are the parent classes for Class C.

## PHP PROGRAMMING

---



Syntax for Multiple Inheritance

//Base Class

class A

```
{  
    public void fooA()  
    {  
        //TO DO:  
    }  
}
```

//Base Class

class B

```
{  
    public void fooB()  
    {  
        //TO DO:  
    }  
}
```

//Derived Class

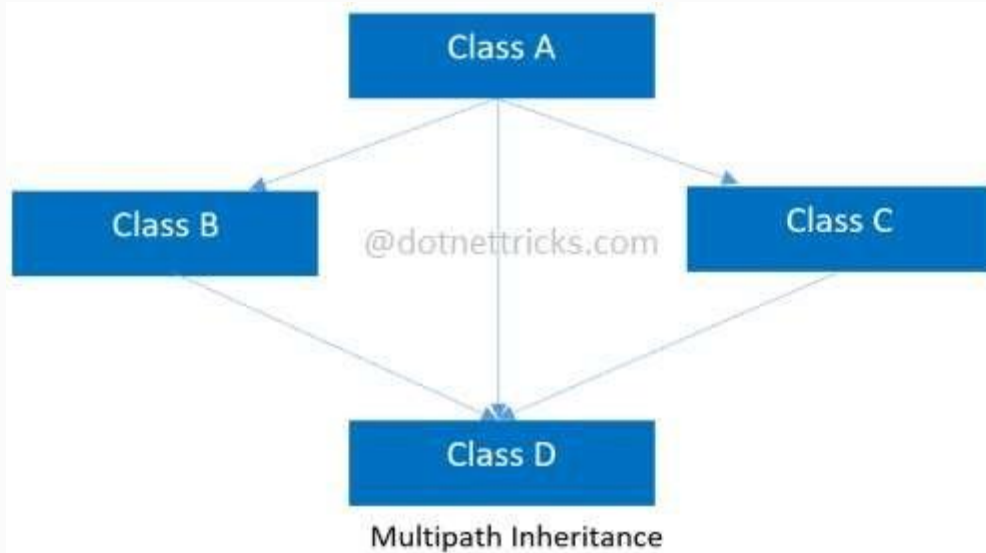
class C : A, B

```
{
```

## PHP PROGRAMMING

```
public void fooC()
{
//TO DO:
}
```

4. Multipath inheritance
5. In this inheritance, a derived class is created from another derived classes and the same base class of another derived classes. This inheritance is not supported by .NET Languages like C#, F# etc.
6. In the given example, class D inherits the properties and behavior of class C and class B as well as Class A. Both class C and class B inherits the Class A. So, Class A is the parent for Class B and Class C as well as Class D. So it's making it Multipath inheritance.
- 7.



Syntax for Multipath Inheritance

//Base Class

class A

```
{
    public void fooA()
    {
```

## PHP PROGRAMMING

---

```
//TO DO:
}
}

//Derived Class
class B : A
{
    public void fooB()
    {
        //TO DO:
    }
}

//Derived Class
class C : A
{
    public void fooC()
    {
        //TO DO:
    }
}

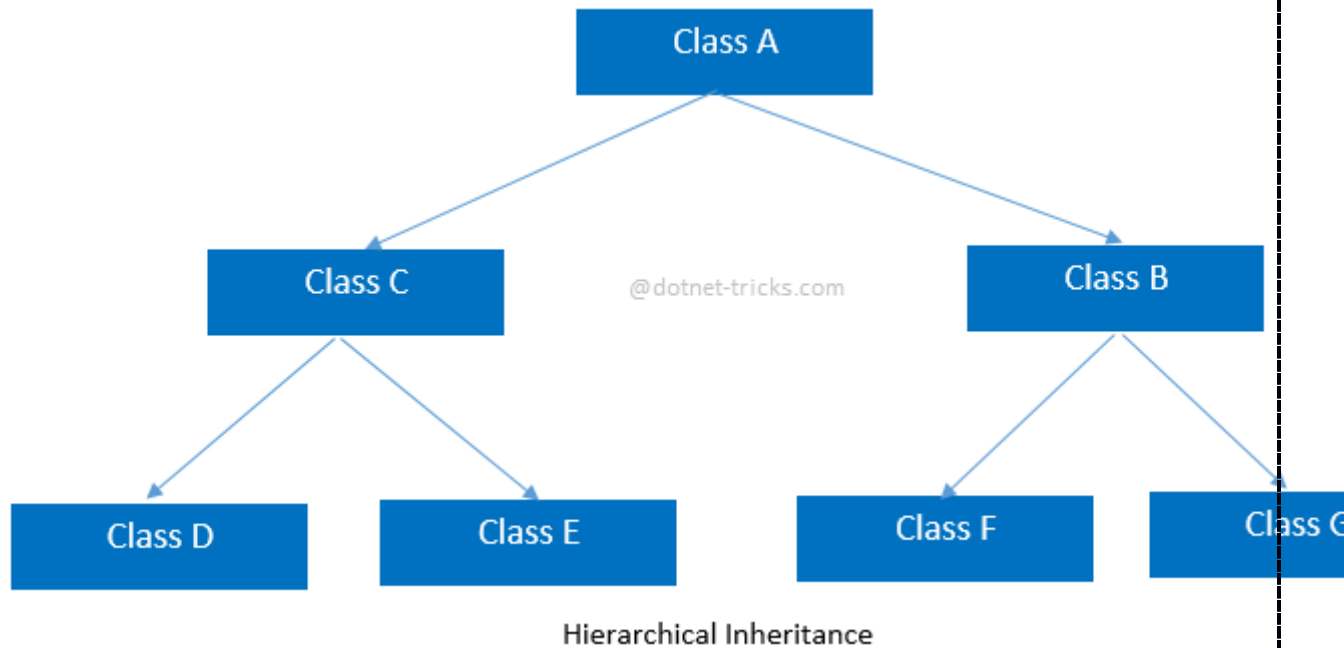
//Derived Class
class D : B, A, C
{
    public void fooD()
    {
        //TO DO:
    }
}
```

### 8. Hierarchical Inheritance

In this inheritance, more than one derived classes are created from a single base class and further child classes act as parent classes for more than one child classes.

## PHP PROGRAMMING

In the given example, class A has two children class B and class C. Further, class B and class C both are having two children - class D and E; class F and G respectively.



Syntax for Hierarchical Inheritance

```
//Base Class
class A
{
    public void fooA()
    {
        //TO DO:
    }
}

//Derived Class
class B : A
{
    public void fooB()
    {
```

## PHP PROGRAMMING

---

```
//TO DO:
}
}

//Derived Class
class C : A
{
    public void fooC()
    {
        //TO DO:
    }
}

//Derived Class
class D : C
{
    public void fooD()
    {
        //TO DO:
    }
}

//Derived Class
class E : C
{
    public void fooE()
    {
        //TO DO:
    }
}

//Derived Class
class F : B
```



## PHP PROGRAMMING

---

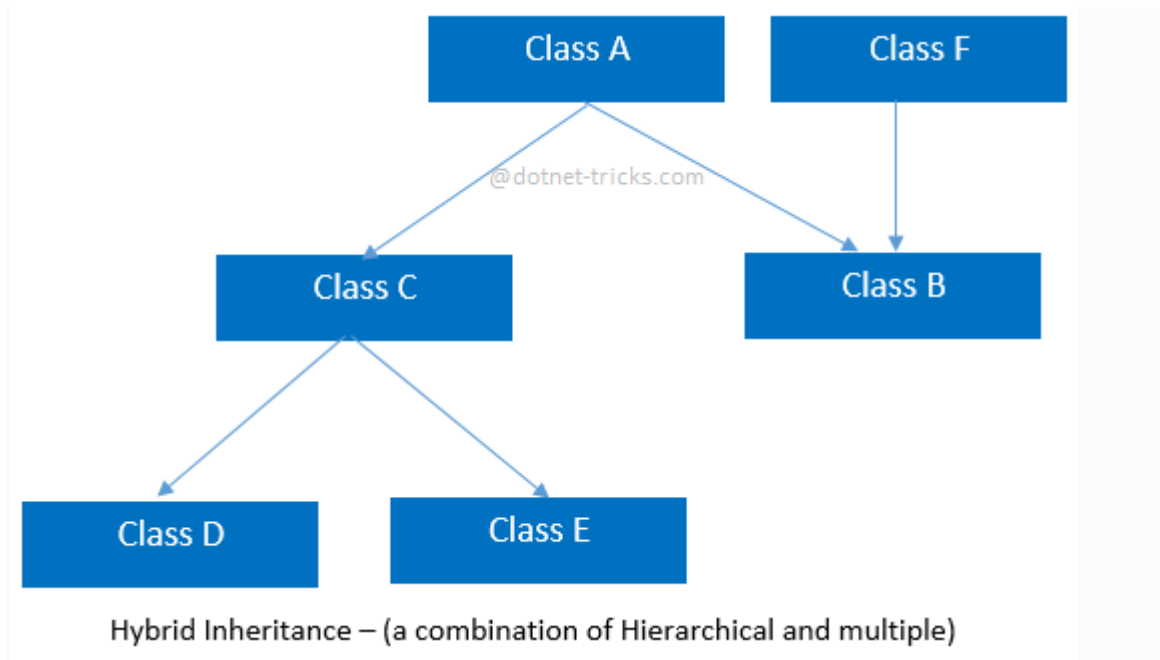
```
{  
    public void fooF()  
    {  
        //TO DO:  
    }  
}  
  
//Derived Class  
class G :B  
{  
    public void fooG()  
    {  
        //TO DO:  
    }  
}
```

### 9. Hybrid Inheritance

This is combination of more than one inheritance. Hence, it may be a combination of Multilevel and Multiple inheritance or Hierarchical and Multilevel inheritance or Hierarchical and Multipath inheritance or Hierarchical, Multilevel and Multiple inheritance.

Since .NET Languages like [C#](#), F# etc. does not support multiple and multipath inheritance. Hence hybrid inheritance with a combination of multiple or multipath inheritances is not supported by .NET Languages.

## PHP PROGRAMMING



Syntax for Hybrid Inheritance

//Base Class

class A

{

public void fooA()

{

//TO DO:

}

}

//Base Class

**Prepared By: Dept Of CSE, RGM CET      Page 18**

Creating an object from an existing class is called instantiating the object

## PHP PROGRAMMING

---

```
class F
{
    public void fooF()
    {
        //TO DO:
    }
}
```

//Derived Class

```
class B : A, F
{
    public void fooB()
    {
        //TO DO:
    }
}
```

//Derived Class

```
class C : A
{
    public void fooC()
```

---

**Prepared By: Dept Of CSE, RGM CET      Page 19**

Creating an object from an existing class is called instantiating the object

## PHP PROGRAMMING

---

```
{  
//TO DO:  
}  
}  
  
//Derived Class  
class D : C  
{  
    public void fooD()  
    {  
        //TO DO:  
    }  
}
```

```
//Derived Class  
class E : C  
{  
    public void fooE()  
    {  
        //TO DO:  
    }  
}
```

## PHP PROGRAMMING

---

```
}
```

### Advantages of Inheritance

1. Reduce code redundancy.
2. Provides code reusability.
3. Reduces source code size and improves code readability.
4. The code is easy to manage and divided into parent and child classes.
5. Supports code extensibility by overriding the base class functionality within child classes.

### **SUPPORTING FUNCTIONS IN PHP TO GET INFORMATION ABOUT CLASS AND OBJECT:**

#### **get\_class():**

Using this function, we can get class name of an object.

```
<?php
    class cls
    {
    }
    $obj=new cls();
    echo get_class($obj);
?>
```

## PHP PROGRAMMING

---

### **class\_exist():**

To check input class is existing or not in current script.

```
<?php
    class cls
    {
    }
    echo class_exist("cls");
?>
```

### **get\_declared\_classes():**

To get all declared classes of current script with predefined classes. It returns output in the form of array.

```
<?php
    class cls1
    {
    }
    class cls2
    {
    }
    print_r(get_declared_classes());
?>
```

### **method\_exist():**

To check input method exists or not in specified class.

```
<?php
    class cls
    {
```

## PHP PROGRAMMING

---

```
function fun()
{
}

}

echo method_exists("cls","fun");

?>
```

### **property\_exists():**

To check input property is existed or not in specified class.

```
<?php
class cls
{
    var $x=10;
}

echo property_exists("cls","x");

?>
```

### **get\_class\_vars():**

### **get\_class\_methods():**

### **get\_object\_vars():**

### **interface\_exists():**

### **is\_subclass\_of():**

### **get\_parent\_class():**

### **return:**

### **\$this:**

### **static:**

# PHP PROGRAMMING

---

**scope resolution operator:**

**parent:**

**access specifiers:**

<https://www.brainbell.com/tutorials/php/abstract-interface-composition-aggregation.html>

<https://www.valuebound.com/resources/blog/object-oriented-programming-concepts-php-part-1>

<https://www.dyclassroom.com/php/php-oop-interface>

<https://www.phptpoint.com/php-polymorphism/>

<https://www.phptpoint.com/php-inheritance/>

**Access Modifier** allows you to alter the visibility of any class member(properties and method).

**In php there are three scopes for class members.**

- Public
- Protected and
- Private

## **Public Access modifier:**

Public access modifier is open to use and access inside the class definition as well as outside the class definition.

```
<?php
```

```
class demo
```

```
{
```

```
    public $name="ravish";
```

```
    function disp()
```



## PHP PROGRAMMING

---

```
        {  
            echo $this->name."<br/>";  
        }  
    }  
class child extends demo  
{  
    function show()  
    {  
        echo $this->name;  
    }  
}  
$obj= new child;  
echo $obj->name."<br/>";  
$obj->disp();  
$obj->show();  
?>
```

**Output:**ravish

ravish

ravish

### Protected access modifier

Protected is only accessible within the class in which it is defined and its parent or inherited classes.

```
<?php
```

```
class demo
```

```
{
```

## PHP PROGRAMMING

---

```
protected $x=500;
protected $y=500;
function add()
{
    echo $sum=$this->x+$this->y."<br/>";
}
}
class child extends demo
{
    function sub()
    {
        echo $sub=$this->x-$this->y."<br/>";
    }
}
$obj= new child;
$obj->add();
$obj->sub();
?>
```

**Output:**1000

0

### Private access modifier

Private is only accessible within the class that defines it.( it can't be access outside the class means in inherited class).

```
<?php
```

```
class demo
```

```
{
```

## PHP PROGRAMMING

---

```
private $name="shashi";
private function show()
{
    echo "This is private method of parent class";
}
}
class child extends demo
{
    function show1()
    {
        echo $this->name;
    }
}
$obj= new child;
$obj->show();
$obj->show1();
?>
```

**Output:Fatal error: Call to private method demo::show()**

### use of all access modifier in a single example

Create a parent class with public, private and protected properties and try to access these properties with their child class.

```
<?php
class parents
{
    public $name="shashi";
    protected $profile="developer";
```

## PHP PROGRAMMING

---

```
private $salary=50000;
public function show()
{
    echo "Welcome : ".$this->name."<br/>";
    echo "Profile : ".$this->profile."<br/>";
    echo "Salary : ".$this->salary."<br/>";
}
}
class childs extends parents
{
    public function show1()
    {
        echo "Welcome : ".$this->name."<br/>";
        echo "Profile : ".$this->profile."<br/>";
        echo "Salary : ".$this->salary."<br/>";
    }
}
$obj= new childs;
$obj->show1();
?>
```

### Output

Welcome:shashi  
Profile:developer  
Salary :

## **PHP PROGRAMMING**

---

### **UNIT V**

**PHP ADVANCED CONCEPTS-** USING COOKIES, USING HTTP HEADERS, USING SESSIONS, USING ENVIRONMENT AND CONFIGURATION VARIABLES.

**WORKING WITH DATE AND TIME-**DISPLAYING HUMAN-READABLE DATES AND TIMES, FINDING THE DATE FOR A WEEKDAY, GETTING THE DAY AND WEEK OF THE YEAR, DETERMINING WHETHER A GIVEN YEAR IS A LEAP YEAR, OBTAINING THE DIFFERENCE BETWEEN TWO DATES, DETERMINING THE NUMBER OF DAYS IN THE CURRENT MONTH, DETERMINING THE NUMBER OF DAYS IN ANY GIVEN MONTH.

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

**OR**

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

**OR**

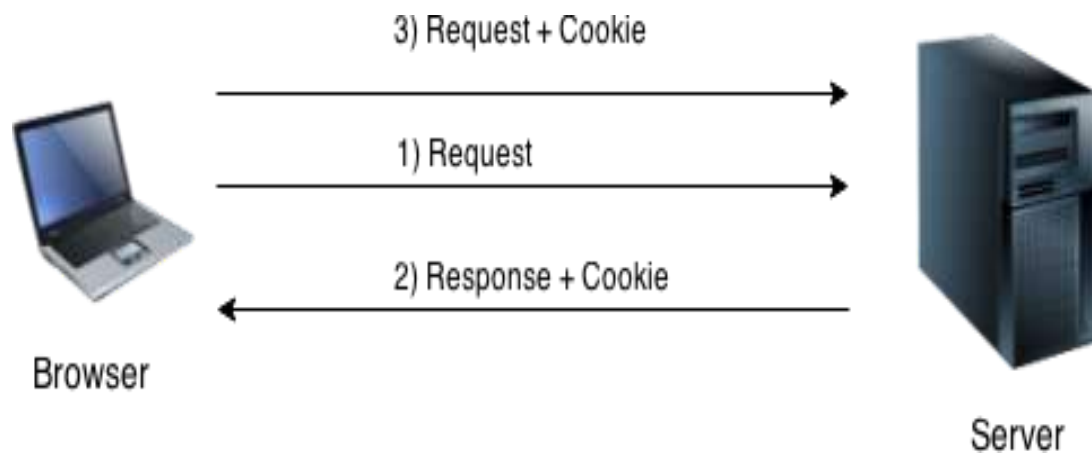
Cookies are text files stored on the client computer and they are kept of use tracking purpose

## PHP PROGRAMMING

---

There are three steps involved in identifying returning users –

- ➔ Server script sends a set of cookies to the browser. For ex name, age, or identification number etc.
- ➔ Browser stores this information on local machine for future use.
- ➔ When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.



### Create Cookies With PHP

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

## PHP PROGRAMMING

---

**Syntax** `setcookie ( string name [, string value [, int expire [, string path [, string domain [, bool secure]]]] )`

Parameter	Description
<ul style="list-style-type: none"><li>name</li></ul>	The name to set the cookie variable to and hence the name to access it with
<ul style="list-style-type: none"><li>value</li></ul>	The value of the current cookie
<ul style="list-style-type: none"><li>expire</li></ul>	When a cookie will expire (in the form of a Unix timestamp)
<ul style="list-style-type: none"><li>path</li></ul>	The directory where the cookie will be available for use
<ul style="list-style-type: none"><li>domain</li></ul>	The domain at which the cookie will be available
<ul style="list-style-type: none"><li>secure</li></ul>	Whether a cookie can be read on a non-SSL enable

script

```
setcookie($cookie_name, $cookie_value, time() + (86400 * 30));
```

```
<?php
```

```
    setcookie("username", "John Carter", time()+30*24*60*60);
```

```
?>
```

## PHP PROGRAMMING

---

**Retrieve a Cookie:** We then retrieve the value of the cookie "user" (using the global variable \$\_COOKIE). We also use the isset() function to find out if the cookie is set or not:

```
<?php
    echo $_COOKIE["username"];

?>

<?php
    if(isset($_COOKIE["username"]))
        echo "Hi " . $_COOKIE["username"];
    else
        echo "Welcome Guest!";

?>
```

**Modify a Cookie Value:** To modify a cookie, just set (again) the cookie using the setcookie() function:

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

?>
```



## PHP PROGRAMMING

---

**Deleting Cookie with PHP:** You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need to set the expiration date in the past, as shown in the example below:

```
<?php  
    setcookie("username", "", time()-3600);  
?>
```

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer.

### **What is a PHP Session?**

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state. Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser. So, Session variables hold information about one single user, and are available to all pages in one application.

### **Why and when to use Sessions?**

## PHP PROGRAMMING

---

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

**Creating a Session:** A session is started with the `session_start()` function.

```
<?php  
    session_start();
```

```
?>
```

**Creating php session variables:** Session variables are set with the PHP global variable: `$_SESSION`.

```
<?php  
    $_SESSION["favcolor"] = "green";  
    $_SESSION["favanimal"] = "cat";
```

## PHP PROGRAMMING

---

?>

### Get PHP Session Variable Values

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

<?php

```
echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";  
echo "Favorite animal is " . $_SESSION["favanimal"] . " .";
```

?>

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

<?php

```
print_r($_SESSION);
```

?>

O/P: Array ( [favcolor] => green [favanimal] => cat )

### Modify a PHP Session Variable

To change a session variable, just overwrite it:

<?php

```
$_SESSION["favcolor"] = "yellow";  
print_r($_SESSION);
```

## PHP PROGRAMMING

---

?>

O/P: Array ( [favcolor] => yellow [favanimal] => cat )

### Destroy a PHP Session

To remove all global session variables and destroy the session,

use `session_unset()` and `session_destroy()`:

<?php

// remove all session variables

session\_unset();

// destroy the session

session\_destroy();

echo "session are unset";

?>

O/P: session are unset

### PHP header() Function

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

---

## PHP PROGRAMMING

---

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

```
<html>
```

```
<?php
```

```
header('Location: http://www.example.com/');
```

```
?>
```

Syntax: header(string,replace,http\_response\_code)

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

## PHP PROGRAMMING

---

### Using HTTP Headers

The most important aspect to remember about headers is that they can be called only before any output has been written to the web page. If you attempt to call a header after output has been sent to the page, you will generate an error; hence, your script will fail on you.

You can use them to control everything, **including setting the current page location, finding out what file format is being displayed, and managing all aspects of the browser cache.** The header() function's prototype is as follows:

```
void header ( string string [, bool replace [, int http_response_code]] )
```

### Redirecting to a Different Location

One of the more common uses for HTTP headers is **redirecting a script**. By using headers inside processing scripts, you can force the browser to return to any page you want. We prefer to use headers to control exception handling within process scripts. The following script makes sure that all input coming from a form is not blank.

```
<html>
<body>
<form action="sample12_5.php" method="post">
Name: <input type="text" name="yourname" ><br />
<input type="submit" value="Submit" >
```

## PHP PROGRAMMING

---

```
</form>
```

```
</body>
```

```
</html>
```

The form in the previous block of code will then call the processing statement as follows:

```
<?php
```

```
//You will assume that this scripts main focus is to validate against a blank entry.
```

```
if (trim ($_POST['yourname']) == '')
```

```
{
```

```
    header ("Location: sample12_5.html");
```

```
    exit;
```

```
}
```

```
echo $_POST['yourname'];
```

```
?>
```

The header() function is rather nice in that it will redirect you automatically to the appropriate file (providing it exists) without a single hiccup in the processing. You will simply find yourself at the appropriate page. You can even use the header() function with the Location parameter to send you to a page not currently on the server on which the script is located.

### **Sending Content Types Other Than HTML**

## PHP PROGRAMMING

---

Naturally, sometimes you will want to use the header() function to output a type of file format that may not be an actual web page. Thankfully, the header function is more than versatile enough to take care of this issue. To make the most out of this function, you can effectively output other file types by simply declaring the content type you want to output. This functionality can be handy in circumstances where you want to deploy a document to a user or perhaps even output a dynamic image. You can use the following script to output a JPG image to the user.

```
<html>
<body>

</body>
</html>

<?php
$path = "images/winter.jpg";
try
{
    if (is_file ($path))
    {
```



## PHP PROGRAMMING

---

```
if ($file = fopen($path, 'rb'))
{
    while(!feof($file) and (connection_status()==0))
    {
        $f .= fread($file, 1024*8);
    }
    fclose($file);
}
//Use the header function to output an image of .jpg.
header ("Content-type: image/jpeg");
print $f;
}
else
{
    throw new exception ("Sorry, file path is not valid.");
}
}
catch (exception $e)
{
```

## PHP PROGRAMMING

---

```
//Create a dynamic error message.  
$animage = imagecreate (500, 500);  
$red = imagecolorallocate ($animage, 255, 0, 0);  
$white = imagecolorallocate ($animage, 255, 255, 255);  
imagefilledrectangle ($animage, 0, 0, 500, 500, $white);  
imagestring ($animage,4,((500-(strlen($e->getMessage()))  
* imagefontwidth(4))) / 2), 5, $e->getMessage(), $red);  
imagejpeg ($animage);  
header ("Content-type: image/jpeg");  
imagedestroy ($animage);  
}  
?>
```

Although the error handling for this particular function may be a tad beyond the scope of this particular chapter, those who have studied Chapter 8 should have no trouble with it. Exception handling aside, what you are doing here is basically reading a file as a binary object. Then, by utilizing the header() function, you can output it as a JPG by merely printing it. You can use this same sort of procedure to read pretty much any file as a binary object and then output it in much

## PHP PROGRAMMING

---

the same way, provided you use the proper content type (more widely known as a MIME type). Table lists a few of the popular MIME types you may be interested in using as output.

Content Type	Application
application/pdf	Adobe Portable Document Format (PDF) types
application/msword	Microsoft Word documents
application/excel	Microsoft Excel documents
image/gif	GIF images
image/png	PNG images
application/octet-stream	Zip files
text/plain	Plain text (text files)

### USING ENVIRONMENT AND CONFIGURATION VARIABLES:

PHP provides a way to use and verify the configuration settings and environment variables relative to the server space the script is occupying. By having access to environment variables, you can customize your scripts to work optimally on the platform that is available. By having access to the configuration variables of PHP, you can customize the PHP environment your script is working in for special occurrences.

## PHP PROGRAMMING

---

A common use of the environment variables in PHP is for dynamic imaging. By using PHP's environment variables to determine the current operating system, you can make your code slightly more portable.

Using configuration variables in PHP is for particularly with file upload scripts.

The PHP installation leaves only enough processing time to upload files that are generally 2MB or smaller in size. By manipulating the PHP configuration files temporarily, you can increase the limit enough to allow a script to process much larger files.

### **Reading Environment and Configuration Variables**

PHP 5 makes reading environment and configuration variables easy. The `$_ENV` superglobal is used for reading a system's environment variables and has an argument set that is based upon the current environment that is available to it. Because of its flexibility, there is no real set argument list, as it is generated based on the current server environment. You can use the `phpinfo()` function to determine the current environment variables, and you can retrieve them using the `getenv()` function, which needs to be supplied a valid environment variable name.

Reading configuration variables, on the other hand, takes place through two functions, `ini_get()` and `ini_get_all()`. The function `ini_get()` will retrieve the value of a specified configuration variable, and the function `ini_get_all()` will retrieve an array filled with the entire selection of configuration variables that are available.

## PHP PROGRAMMING

---

**Example shows how to retrieve both environment and configuration variables.**

**<?php**

//Here is an example of retrieving an environmental variable or two.

echo \$\_ENV['ProgramFiles'] . "<br />"; //Outputs C:\Program Files.

echo \$\_ENV['COMPUTERNAME'] . "<br />"; //Outputs BABINZ-CODEZ.

echo getenv("COMPUTERNAME") . "<br />"; //Also Outputs

//Now, let's look at reading configuration variables.

echo ini\_get ("post\_max\_size") . "<br />"; //Outputs 8MB.

//And you can output the entire listing with this function.

print\_r (ini\_get\_all());

**?>**

### **Setting Environment and Configuration Variables**

Setting environment and configuration variables is just as easy as it is to get them. While working with environment variables, you must need to assign a new value to the \$\_ENV superglobal to process a temporary change. The change will be in effect for the script's duration. The same applies for configuration variables but with a different approach. To set a configuration variable, you have to use the PHP function ini\_set(), which will allow you to set a

## PHP PROGRAMMING

---

configuration variable for the script's duration. Once the script finishes executing, the configuration variable will return to its original state. The prototype for `ini_set()` is as follows:

**string ini\_set ( string varname, string newvalue )**

**<?php**

//Setting an environment variable in php is as easy as assigning it.

echo \$\_ENV['COMPUTERNAME'] . "<br />"; // Echoes BABINZ-CODEZ.

\$\_ENV['COMPUTERNAME'] = "Hello World!";

echo \$\_ENV['COMPUTERNAME'] . "<br />"; //Echoes the new COMPUTERNAME.

//Of course the change is relevant only for the current script.

//Setting a configuration variable is the same in that it is in effect only for the duration of the script.

echo ini\_get ('post\_max\_size'); //Echoes 8MB.

//Then you set it to 200M for the duration of the script.

ini\_set('post\_max\_size','200M');

//Any files that are to be uploaded in this script will be OK up to 200M.

**?>**

## DATE & TIME FUNCTIONS

Function	Description
----------	-------------

## PHP PROGRAMMING

---

<b>checkdate()</b>	Validates set of Gregorian year, month, and day values (for example, 2005, 3, 17).
<b>date_sunrise()</b>	Returns time of sunrise for a given day and location (new in PHP 5).
<b>date_sunset()</b>	Returns time of sunset for a given day and location (new in PHP 5).
<b>date()</b>	Formats a local date/time, given a Unix timestamp (for example, 1111035030000 ) and a formatting string.
<b>getdate()</b>	Given a Unix timestamp, returns an associative array containing date and time information (defaults to current time).
<b>gettimeofday()</b>	Returns an associative array containing information about the current system time.
<b>gmdate()</b>	Formats a GMT/UTC date/time. Uses the same formatting characters as the date() function.
<b>gmmktime()</b>	Converts a set of GMT date/time values into a Unix timestamp (analogous to mktime()).
<b>gmstrftime()</b>	Formats a GMT/UTC date/time according to locale settings
<b>idate()</b>	Formats a local time/date value as an integer.
<b>localtime()</b>	Given a Unix timestamp, returns an array of date/time values..

## PHP PROGRAMMING

---

<b>microtime()</b>	Returns a string representation of the current Unix timestamp with microseconds.
<b>mktime()</b>	Converts a set of local date/time values into a Unix timestamp.
<b>strftime()</b>	Given a timestamp and a formatting string, returns a representation of a local date/time according to locale settings.
<b>strtotime()</b>	Given a date/time string generated with strftime() and the formatting string used to generate it, returns a Unix timestamp (new in PHP 5.1).
<b>strtotime()</b>	Converts an English textual date/time description into a Unix timestamp.
<b>time()</b>	Returns the current system date and time as a Unix timestamp.

### Formatting Characters for the date() Function

Character	Description
<b>Month</b>	
<b>F</b>	Full name of the month (January, February, and so on).
<b>M</b>	Three-letter abbreviation for the month (Jan, Feb, and so on).
<b>m</b>	Numeric representation for the month, with leading zero (two digits).
<b>n</b>	Numeric representation for the month (no leading zero).



## PHP PROGRAMMING

---

### Day

<b>d</b>	Day of the month, with leading zeros (two digits).
<b>j</b>	Day of the month (no leading zeros).
<b>S</b>	Ordinal suffix for the day of the month, two characters (st, nd, th); most commonly used in combination with j.
<b>l</b> (lowercase L)	Full name of the day of the week (Monday, Tuesday, and so on).
<b>D</b>	A textual representation of a day, three letters (Mon, Tue, and so on).
<b>w</b>	Numeric representation of the day of the week (0 = Sunday, 6 = Saturday).

### Year

<b>y</b>	Two-digit year.
<b>Y</b>	Four-digit year.

### Hour

<b>h</b>	Hour in 12-hour format, with leading zero (two digits).
<b>g</b>	Hour in 12-hour format (no leading zero).
<b>H</b>	Hour in 24-hour format, with leading zero (two digits).

## PHP PROGRAMMING

---

<b>G</b>	Hour in 24-hour format (no leading zero).
<b>a</b>	am/pm (lowercase).
<b>A</b>	AM/PM (uppercase).
<b>O</b> (uppercase o)	String representation of the difference in hours between local time and GMT/UTC (for example, +1000, -0500).

### Minute

<b>i</b>	Minute, with leading zero (two digits).
<b>j</b>	Minute (no leading zero).

### Second

<b>s</b>	Second, with leading zero (two digits).
<b>Z</b>	Integer representation of the difference in seconds between local time and GMT/UTC (for example, 36000 for GMT+1000 and -18000 for GMT-0500).

### Complete Date and Time

<b>c</b>	ISO-8601 format (YYYY-MM-DDTHH:MM:SS±HHMM, for example,
----------	---

## PHP PROGRAMMING

---

2005-03-14T19:38:08+10:00).

**r** RFC-2822 format WWW, DD MMM YYYY HH:MM:SS ±HHMM, for example, Mon,14 Mar 2005 19:38:08 +1000).

**U** Seconds since the Unix epoch. Calling date('U') with no timestamp argument produces the same output as the time() function.

## PHP PROGRAMMING

---

### Displaying Human-Readable Dates and Times

❖ To obtain a timestamp for the current system date and time, it is necessary only to call the `time()` function, as shown here:

```
<?php
    echo time();
?>
```

In a web browser, this produces output such as the following:

**1110638611**

❖ For obtaining a human-readable date and time, PHP provides the `date()` function. When called with a single argument (a formatting string), this function returns a string representation of the current date and/or time. The optional second argument is a timestamp.

#### Example1:

```
<?php
    $time = time();
    $formats = array(
        'U',
        'r',
```

## PHP PROGRAMMING

---

```
'c',
'l, F jS, Y, g:i A',
'H:i:s D d M y',
);
foreach($formats as $format)
    echo "<p><b>$format</b>: " . date($format, $time) . "</p>\n";
?>
```

### Output:

```
U: 1110643578
r: Sun, 13 Mar 2005 02:06:18 +1000
c: 2005-03-13T02:06:18+10:00
l, F jS, Y, g:i A: Sunday, March 13th, 2005, 2:06 AM
H:i:s D d M y: 02:06:18 Sun 13 Mar 05
```

### Finding the Date for a Weekday

- ❖ By combining date() and strtotime(), it is possible to get the day for any desired weekday in a given month.

## PHP PROGRAMMING

---

- ❖ For example, suppose that your firm's sales employees are supposed to turn in their monthly sales reports on the first Tuesday of each month. The following example shows how you can determine the date of the first Tuesday in the month following the current one.

### Example2:

```
<?php
    $nextmonth = date('Y-' . (date('n') + 1) . '-01');
    $nextmonth_ts = strtotime($nextmonth);
    $firsttue_ts = strtotime("Tuesday", $nextmonth_ts);
    echo 'Today is ' . date('d M Y') . '.<br />\n';
    echo 'The first Tuesday of next month is ' . date('d M Y', $firsttue_ts) . '.';
?>
```

### Output:

Today is 15 Mar 2005.

The first Tuesday of next month is 05 Apr 2005.

### Explanation:

```
$nextmonth = date('Y-' . (date('n') + 1) . '-01');
```

## PHP PROGRAMMING

---

The inner call to `date()` returns an integer corresponding to the current month, to which you add 1. You then use this as part of the argument to another `date()` call, which returns the string 2005-4-01.

```
$nextmonth_ts = strtotime($nextmonth);:
```

This stores the timestamp equivalent to 2005-4-01 in `$nextmonth_ts`.

```
$firsttue_ts = strtotime("Tuesday", $nextmonth_ts);:
```

Using the timestamp just obtained as the second argument to `strtotime()`, you get a new timestamp, `$firsttue_ts`. Since the first argument to `strtotime()` is simply the string Tuesday, the function looks for the first date following the date corresponding to `$nextmonth_ts` that falls on a Tuesday. The timestamp corresponding to this date is stored as `$firsttue_ts`.

```
echo 'Today is ' . date('d M Y') . ' .<br />\n';:
```

To provide a basis for comparison, you output the current date in dd-MM-yyyy format.

```
echo 'The first Tuesday of next month is ' . date('d M Y', $firsttue_ts) . ' .';:
```

## PHP PROGRAMMING

---

Finally, you feed the `$firsttue_ts` timestamp to `date()` and output the result in dd-MM-yyyy format.



## PHP PROGRAMMING

---

### Example3:

```
<?php
    echo 'Today is ' . date('d M Y') . '.';
    for($i = 1; $i <= 12; $i++)
    {
        $nextmonth = date('Y-' . (date('n') + $i) . '-01');
        $nextmonth_ts = strtotime($nextmonth);
        $firsttue_ts = strtotime("Tuesday", $nextmonth_ts);
        echo "\n<br />The first Tuesday in ' . date('F', $firsttue_ts)
        . ' is ' . date('d M Y', $firsttue_ts) . '.';
    }
?>
```

## PHP PROGRAMMING

---

### Output:

Today is 15 Mar 2005.  
The first Tuesday of April is 05 Apr 2005.  
The first Tuesday of May is 03 May 2005.  
The first Tuesday of June is 07 Jun 2005.  
The first Tuesday of July is 05 Jul 2005.  
The first Tuesday of August is 02 Aug 2005.  
The first Tuesday of September is 06 Sep 2005.  
The first Tuesday of October is 04 Oct 2005.  
The first Tuesday of November is 01 Nov 2005.  
The first Tuesday of December is 06 Dec 2005.  
The first Tuesday of January is 03 Jan 2006.  
The first Tuesday of February is 07 Feb 2006.  
The first Tuesday of March is 07 Mar 2006.

## PHP PROGRAMMING

---

### Getting the Day and Week of the Year

- ❖ Obtaining the day of the year is fairly simple; you need use only a lowercase *z* in the first argument to the date() function.

#### Example4:

```
<?php
    $mydates = array('2005-01-01', '2005-06-30', '2005-12-31');
    foreach($mydates as $mydate)
    {
        $ts = strtotime($mydate);
        echo 'Day ' . date('d M Y: z', $ts) . "<br />\n";
    }
?>
```

#### Output:

```
01 Jan 2005: Day 0
30 Jun 2005: Day 180
31 Dec 2005: Day 364
```

## PHP PROGRAMMING

---

**NOTE:** The numbering of the days of the year as derived using date('z') begins with 0, which means you will likely need to add 1 to the result before displaying it.

- ❖ Getting the number of the week in the year is also quite simple: all that is required is to pass an uppercase letter as a formatting character to date().

### Example5:

```
<?php
    $mydates = array('2005-01-01', '2005-01-03', '2005-05-22', '2005-05-23',
    '2005-12-31');
    foreach($mydates as $mydate)
        echo date("D d M Y: \w\e\e\k W", strtotime($mydate)) . "<br />\n";
?>
```

### Output:

```
Sat 01 Jan 2005: week 53
Mon 03 Jan 2005: week 1
Sun 22 May 2005: week 20
```

## **PHP PROGRAMMING**

---

Mon 23 May 2005: week 21

Sat 31 Dec 2005: week 52

## PHP PROGRAMMING

---

### Determining Whether a Given Year Is a Leap Year

- ❖ The `date()` function employs another one-letter argument; it uses *L* to determine if a given year is a leap year.
- ❖ When *L* is used, `date()` returns 1 if the year in question is a leap year and 0 if it is not.
- ❖ Rather than make repeated calls to `date()` and `strtotime()`, you can wrap this in a simple function that takes the year to be tested as an argument, as shown in the following example.

#### Example6:

```
<?php
```

## PHP PROGRAMMING

---

```
// takes a 2- or 4-digit year,
// returns 1 or 0
function is_leap_year($year)
{
    $ts = strtotime("$year-01-01");
    return date('L', $ts);
}

// test the function for a set of 11 consecutive years
for($i = 2000; $i <= 2010; $i++)
{
    $output = "$i is ";
    if( !is_leap_year($i) )
        $output .= "not ";
    $output .= "a leap year.<br />\n";
    echo $output;
}

?>
```

### How It Works

## PHP PROGRAMMING

---

The result of the test loop is as follows:

2000 is a leap year.

2001 is not a leap year.

2002 is not a leap year.

2003 is not a leap year.

2004 is a leap year.

2005 is not a leap year.

2006 is not a leap year.

2007 is not a leap year.

2008 is a leap year.

2009 is not a leap year.

2010 is not a leap year.

**NOTE:** A final note regarding leap years: you should remember that years ending in 00 are leap years *only* if the first two digits of the year taken as a two-digit number are evenly divisible by 4. This means that although 2000 was a leap year ( $20 \% 4 = 0$ ), 1900 and 2100 are not ( $19 \% 4 = 3$ ;  $21 \% 4 = 1$ ).



## **PHP PROGRAMMING**

---

### **Obtaining the Difference between Two Dates**

## PHP PROGRAMMING

---

As you have already had the chance to see, altering a date by a given interval is not difficult.

Getting the difference between two dates is a bit more complicated.

### **Example7:**

```
<?php
```

```
    $date1 = '14 Jun 2002';
```

```
    $date2 = '05 Feb 2006';
```

```
    $ts1 = strtotime($date1);
```

```
    $ts2 = strtotime($date2);
```

```
    $min = ($ts2 - $ts1)/60;
```

```
    $hour = $min/60;
```

```
    $day = $hour/24;
```

```
    printf("<p>The difference between %s and %s is %d seconds.<p>\n",$date1, $date2, $ts2  
        - $ts1);
```

```
    printf("<p>The difference between %s and %s is %d minutes.<p>\n",$date1, $date2  
        , $min);
```

```
    printf("<p>The difference between %s and %s is %d hours.<p>\n",$date1, $date2,  
        $hour);
```

```
    printf("<p>The difference between %s and %s is %d days.<p>\n",$date1, $date2, $day);
```

## PHP PROGRAMMING

---

?>

### Output:

The difference between 14 Jun 2002 and 05 Feb 2006 is 115084800 seconds.

The difference between 14 Jun 2002 and 05 Feb 2006 is 1918140 minutes.

The difference between 14 Jun 2002 and 05 Feb 2006 is 31969 hours.

The difference between 14 Jun 2002 and 05 Feb 2006 is 1332 days.

### Determining the number of days in the current month

- ❖ We are using date('t') function for finding number of days in a month.
- ❖ For finding number of days in current months we use both date('t') and date('M') functions.

### Example8:

<?php

## PHP PROGRAMMING

---

```
echo 'Number of days for the month of '.date('M'). ' is :'.date('t')."\n";  
?>
```

### Output:

Number of days for the month of Sep is: 30

### Determining the number of days in any given month

- ❖ The `cal_days_in_month()` function returns the number of days in a month for a specified year and calendar.
- ❖ Syntax is

```
int cal_days_in_month ( int $calendar , int $month , int $year )
```

### Example9:

```
<?php  
echo cal_days_in_month(CAL_GREGORIAN, 10, 2014);  
echo date('t',strtotime('2014/10/01'));  
?>
```

### Output:

31

## PHP PROGRAMMING

---

31

**Convert the month number to month name.**

**Example:**

```
<?php
    echo 'Write your code here';
    $month_num = 9;
    $dateObj = DateTime::createFromFormat('!m', $month_num);
    $month_name = $dateObj->format('F');
    echo $month_name."\n";
?>
```

**Output:**

September

## PHP PROGRAMMING

---

**Count the number of days between current day and birthday.**

```
<?php
    $target_days = mktime(0,0,0,12,31,2018);// modify the birth day 12/31/2013
    $today = time();
    $diff_days = ($target_days - $today);
    $days = (int)($diff_days/86400);
    print "Days till next birthday: $days days!". "\n";
?>
```

# PHP PROGRAMMING

---

## DATE & TIME FUNCTIONS

| Function              | Description  |
|-----------------------|--|
| <b>checkdate()</b>    | Validates set of Gregorian year, month, and day values (for example, 2005, 3, 17). |
| <b>date_sunrise()</b> | Returns time of sunrise for a given day and location (new in PHP 5).               |
| <b>date_sunset()</b>  | Returns time of sunset for a given day and location (new in PHP 5).                |

## PHP PROGRAMMING

---

|                       |  |
|-----------------------|--|
| <b>date()</b>         | Formats a local date/time, given a Unix timestamp (for example, 1111035030000 ) and a formatting string.               |
| <b>getdate()</b>      | Given a Unix timestamp, returns an associative array containing date and time information (defaults to current time).  |
| <b>gettimeofday()</b> | Returns an associative array containing information about the current system time.                                     |
| <b>gmdate()</b>       | Formats a GMT/UTC date/time. Uses the same formatting characters as the date() function.                               |
| <b>gmmktime()</b>     | Converts a set of GMT date/time values into a Unix timestamp (analogous to mktime()).                                  |
| <b>gmstrftime()</b>   | Formats a GMT/UTC date/time according to locale settings   |
| <b>idate()</b>        | Formats a local time/date value as an integer.   |
| <b>localtime()</b>    | Given a Unix timestamp, returns an array of date/time values..   |
| <b>microtime()</b>    | Returns a string representation of the current Unix timestamp with microseconds.                                       |
| <b>mktime()</b>       | Converts a set of local date/time values into a Unix timestamp.  |
| <b>strftime()</b>     | Given a timestamp and a formatting string, returns a representation of a local date/time according to locale settings. |



## PHP PROGRAMMING

---

|                    |  |
|--------------------|--|
| <b>strtotime()</b> | Given a date/time string generated with strftime() and the formatting string used to generate it, returns a Unix timestamp (new in PHP 5.1). |
| <b>strtotime()</b> | Converts an English textual date/time description into a Unix timestamp.   |
| <b>time()</b>      | Returns the current system date and time as a Unix timestamp.  |

### Formatting Characters for the date() Function

| Character              | Description   |
|------------------------|---|
| <b>Month</b>           |   |
| <b>F</b>               | Full name of the month (January, February, and so on).  |
| <b>M</b>               | Three-letter abbreviation for the month (Jan, Feb, and so on).  |
| <b>m</b>               | Numeric representation for the month, with leading zero (two digits).   |
| <b>n</b>               | Numeric representation for the month (no leading zero).   |
| <b>Day</b>             |   |
| <b>d</b>               | Day of the month, with leading zeros (two digits).  |
| <b>j</b>               | Day of the month (no leading zeros).  |
| <b>S</b>               | Ordinal suffix for the day of the month, two characters (st, nd, th); most commonly used in combination with j. |
| <b>l</b> (lowercase L) | Full name of the day of the week (Monday, Tuesday, and so on).  |

## PHP PROGRAMMING

---

**D** A textual representation of a day, three letters (Mon, Tue, and so on).  
**w** Numeric representation of the day of the week (0 = Sunday, 6 = Saturday).

### Year

**y** Two-digit year.  
**Y** Four-digit year.

### Hour

**h** Hour in 12-hour format, with leading zero (two digits).  
**g** Hour in 12-hour format (no leading zero).  
**H** Hour in 24-hour format, with leading zero (two digits).  
**G** Hour in 24-hour format (no leading zero).  
**a** am/pm (lowercase).  
**A** AM/PM (uppercase).  
**O** (uppercase o) String representation of the difference in hours between local time and GMT/UTC (for example, +1000, -0500).

### Minute

## PHP PROGRAMMING

---

**i** Minute, with leading zero (two digits).

**j** Minute (no leading zero).

### Second

**s** Second, with leading zero (two digits).

**Z** Integer representation of the difference in seconds between local time and GMT/UTC (for example, 36000 for GMT+1000 and –18000 for GMT–0500).

### Complete Date and Time

**c** ISO-8601 format (YYYY-MM-DDTHH:MM:SS±HHMM, for example, 2005-03-14T19:38:08+10:00).

**r** RFC-2822 format WWW, DD MMM YYYY HH:MM:SS ±HHMM, for example, Mon,14 Mar 2005 19:38:08 +1000).

**U** Seconds since the Unix epoch. Calling date('U') with no timestamp argument produces the same output as the time() function.

### Displaying Human-Readable Dates and Times

## PHP PROGRAMMING

---

❖ To obtain a timestamp for the current system date and time, it is necessary only to call the `time()` function, as shown here:

```
<?php  
echo time();  
?>
```

In a web browser, this produces output such as the following:

**1110638611**

❖ For obtaining a human-readable date and time, PHP provides the `date()` function. When called with a single argument (a formatting string), this function returns a string representation of the current date and/or time. The optional second argument is a timestamp.

### **Example1:**

```
<?php  
$time = time();  
$formats = array(  
    'U',  
    'r',  
    'c',
```

## PHP PROGRAMMING

---

```
'l, F jS, Y, g:i A',  
'H:i:s D d M y',  
);  
foreach($formats as $format)  
echo "<p><b>$format</b>: " . date($format, $time) . "</p>\n";  
?>
```

### Output:

```
U: 1110643578  
r: Sun, 13 Mar 2005 02:06:18 +1000  
c: 2005-03-13T02:06:18+10:00  
l, F jS, Y, g:i A: Sunday, March 13th, 2005, 2:06 AM  
H:i:s D d M y: 02:06:18 Sun 13 Mar 05
```

### Finding the Date for a Weekday

- ❖ By combining date() and strtotime(), it is possible get the day for any desired weekday in a given month.
- ❖ For example, suppose that your firm's sales employees are supposed to turn in their monthly sales reports on the first Tuesday of each month. The following example shows

## PHP PROGRAMMING

---

how you can determine the date of the first Tuesday in the month following the current one.

### Example2:

```
<?php
$nextmonth = date('Y-' . (date('n') + 1) . '-01');
$nextmonth_ts = strtotime($nextmonth);
$firsttue_ts = strtotime("Tuesday", $nextmonth_ts);
echo 'Today is ' . date('d M Y') . ' .<br />\n';
echo 'The first Tuesday of next month is ' . date('d M Y', $firsttue_ts) . ' .';
?>
```

### Output:

Today is 15 Mar 2005.

The first Tuesday of next month is 05 Apr 2005.

### Explanation:

```
$nextmonth = date('Y-' . (date('n') + 1) . '-01');
```

## PHP PROGRAMMING

---

The inner call to `date()` returns an integer corresponding to the current month, to which you add 1. You then use this as part of the argument to another `date()` call, which returns the string 2005-4-01.

```
$nextmonth_ts = strtotime($nextmonth);:
```

This stores the timestamp equivalent to 2005-4-01 in `$nextmonth_ts`.

```
$firsttue_ts = strtotime("Tuesday", $nextmonth_ts);:
```

Using the timestamp just obtained as the second argument to `strtotime()`, you get a new timestamp, `$firsttue_ts`. Since the first argument to `strtotime()` is simply the string Tuesday, the function looks for the first date following the date corresponding to `$nextmonth_ts` that falls on a Tuesday. The timestamp corresponding to this date is stored as `$firsttue_ts`.

```
echo 'Today is ' . date('d M Y') . ' .<br />\n';:
```

To provide a basis for comparison, you output the current date in dd-MM-yyyy format.

```
echo 'The first Tuesday of next month is ' . date('d M Y', $firsttue_ts) . ' .';:
```

## PHP PROGRAMMING

---

Finally, you feed the \$firsttue\_ts timestamp to date() and output the result in dd-MM-yyyy format.

### Example3:

```
<?php
echo 'Today is ' . date('d M Y') . '.';
for($i = 1; $i <= 12; $i++)
{
    $nextmonth = date('Y-' . (date('n') + $i) . '-01');
    $nextmonth_ts = strtotime($nextmonth);
    $firsttue_ts = strtotime("Tuesday", $nextmonth_ts);
    echo "\n<br />The first Tuesday in ' . date('F', $firsttue_ts)
    . ' is ' . date('d M Y', $firsttue_ts) . '.';
}
?>
```

### Output:

Today is 15 Mar 2005.

The first Tuesday of April is 05 Apr 2005.

The first Tuesday of May is 03 May 2005.



## PHP PROGRAMMING

---

The first Tuesday of June is 07 Jun 2005.

The first Tuesday of July is 05 Jul 2005.

The first Tuesday of August is 02 Aug 2005.

The first Tuesday of September is 06 Sep 2005.

The first Tuesday of October is 04 Oct 2005.

The first Tuesday of November is 01 Nov 2005.

The first Tuesday of December is 06 Dec 2005.

The first Tuesday of January is 03 Jan 2006.

The first Tuesday of February is 07 Feb 2006.

The first Tuesday of March is 07 Mar 2006.

### Getting the Day and Week of the Year

- ❖ Obtaining the day of the year is fairly simple; you need use only a lowercase z in the first argument to the date() function.

#### Example4:

```
<?php
$mydates = array('2005-01-01', '2005-06-30', '2005-12-31');
foreach($mydates as $mydate)
{
```

## PHP PROGRAMMING

---

```
$ts = strtotime($mydate);  
echo 'Day ' . date('d M Y: z', $ts) . "<br />\n";  
}  
?>
```

### Output:

```
01 Jan 2005: Day 0  
30 Jun 2005: Day 180  
31 Dec 2005: Day 364
```

**NOTE:** The numbering of the days of the year as derived using date('z') begins with 0, which means you will likely need to add 1 to the result before displaying it.

- ❖ Getting the number of the week in the year is also quite simple: all that is required is to pass an uppercase was a formatting character to date().

### Example5:

```
<?php  
$mydates = array('2005-01-01', '2005-01-03', '2005-05-22', '2005-05-23',  
'2005-12-31');
```

## PHP PROGRAMMING

---

```
foreach($mydates as $mydate)
echo date("D d M Y: \w\e\e\k W", strtotime($mydate)) . "<br />\n";
?>
```

### Output:

Sat 01 Jan 2005: week 53  
Mon 03 Jan 2005: week 1  
Sun 22 May 2005: week 20  
Mon 23 May 2005: week 21  
Sat 31 Dec 2005: week 52

### Determining Whether a Given Year Is a Leap Year

- ❖ The date() function employs another one-letter argument; it uses L to determine if a given year is a leap year.
- ❖ When L is used, date() returns 1 if the year in question is a leap year and 0 if it is not.
- ❖ Rather than make repeated calls to date() and strtotime(), you can wrap this in a simple function that takes the year to be tested as an argument, as shown in the following example.

### Example6:

## PHP PROGRAMMING

---

```
<?php
// takes a 2- or 4-digit year,
// returns 1 or 0
function is_leap_year($year)
{
    $ts = strtotime("$year-01-01");
    return date('L', $ts);
}
// test the function for a set of 11 consecutive years
for($i = 2000; $i <= 2010; $i++)
{
    $output = "$i is ";
    If( !is_leap_year($i) )
        $output .= "not ";
    $output .= "a leap year.<br />\n";
    echo $output;
}
?>
```

## PHP PROGRAMMING

---

### How It Works

The result of the test loop is as follows:

2000 is a leap year.

2001 is not a leap year.

2002 is not a leap year.

2003 is not a leap year.

2004 is a leap year.

2005 is not a leap year.

2006 is not a leap year.

2007 is not a leap year.

2008 is a leap year.

2009 is not a leap year.

2010 is not a leap year.

**NOTE:** A final note regarding leap years: you should remember that years ending in 00 are leap years only if the first two digits of the year taken as a two-digit number are evenly divisible by 4. This means that although 2000 was a leap year ( $20 \% 4 = 0$ ), 1900 and 2100 are not ( $19 \% 4 = 3$ ;  $21 \% 4 = 1$ ).

### Obtaining the Difference between Two Dates

## PHP PROGRAMMING

---

- ❖ As you have already had the chance to see, altering a date by a given interval is not difficult.
- ❖ Getting the difference between two dates is a bit more complicated.

### Example7:

```
<?php
$date1 = '14 Jun 2002';
$date2 = '05 Feb 2006';
$ts1 = strtotime($date1);
$ts2 = strtotime($date2);
$min = ($ts2 - $ts1)/60;
$hour = $min/60;
$day = $hour/24;
printf("<p>The difference between %s and %s is %d seconds.<p>\n",$date1, $date2, $ts2
    - $ts1);
printf("<p>The difference between %s and %s is %d minutes.<p>\n",$date1, $date2 , $min);
printf("<p>The difference between %s and %s is %d hours.<p>\n",$date1, $date2,      $hour);
printf("<p>The difference between %s and %s is %d days.<p>\n",$date1, $date2, $day);
?>
```

## PHP PROGRAMMING

---

### Output:

The difference between 14 Jun 2002 and 05 Feb 2006 is 115084800 seconds.

The difference between 14 Jun 2002 and 05 Feb 2006 is 1918140 minutes.

The difference between 14 Jun 2002 and 05 Feb 2006 is 31969 hours.

The difference between 14 Jun 2002 and 05 Feb 2006 is 1332 days.

### Determining the number of days in the current month

- ❖ We are using date('t') function for finding number of days in a month.
- ❖ For finding number of days in current months we use both date('t') and date('M') functions.

### Example8:

```
<?php
echo 'Number of days for the month of '.date('M'). ' is :'.date('t')."\n";
?>
```

### Output:

Number of days for the month of Sep is: 30

### Determining the number of days in any given month

## PHP PROGRAMMING

---

- ❖ The `cal_days_in_month()` function returns the number of days in a month for a specified year and calendar.
- ❖ Syntax is

**`int cal_days_in_month ( int $calendar , int $month , int $year )`**

### **Example9:**

```
<?php
echo cal_days_in_month(CAL_GREGORIAN, 10, 2014);
echo date('t',strtotime('2014/10/01'));
?>
```

### **Output:**

31

31

### **Convert the month number to month name.**

### **Example:**

```
<?php
echo 'Write your code here';

$month_num = 9;

$dateObj = DateTime::createFromFormat('!m', $month_num);
```



## PHP PROGRAMMING

---

```
$month_name = $dateObj->format('F');  
echo $month_name."\n";  
?>
```

### **Output:**

September

### **Count the number of days between current day and birthday.**

```
<?php  
$target_days = mktime(0,0,0,12,31,2018);// modify the birth day 12/31/2013  
$today = time();  
$diff_days = ($target_days - $today);  
$days = (int)($diff_days/86400);  
print "Days till next birthday: $days days!". "\n";  
?>
```

# PHP PROGRAMMING

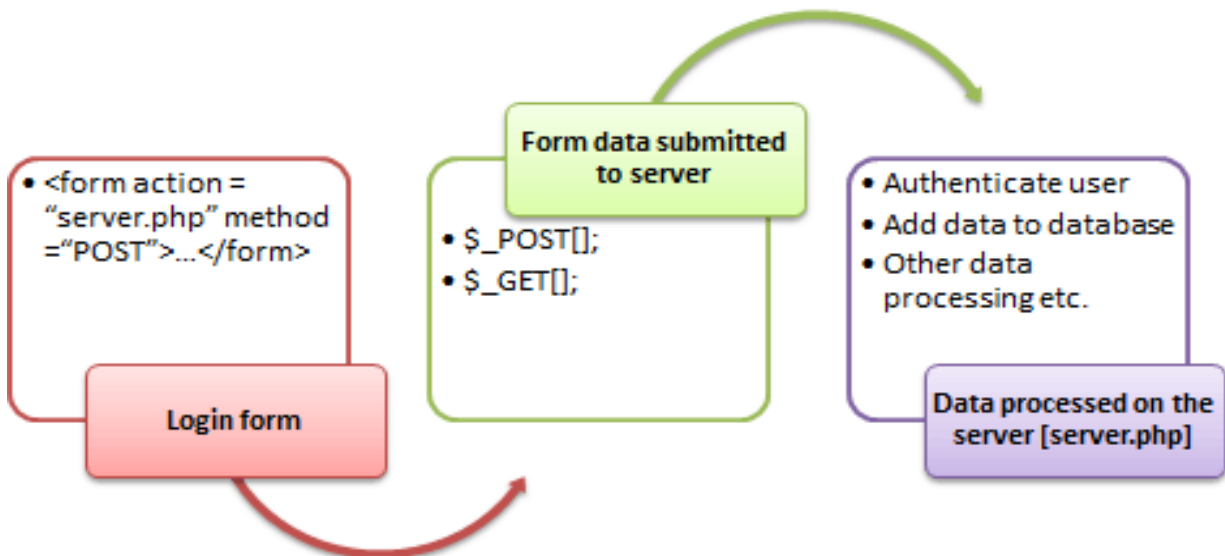
**CREATING AND USING FORMS:** To create a fully functional web application, you need to be able to interact with your users. The common way to receive information from web users is through a form. Generally we have several options when dealing with forms. More specifically, you have control over

- what elements you want to provide to your user,
- how you handle the information passed to you, and
- In what format you choose to receive the data.

Obviously, when dealing with information that is passed from a user, it is required that you spend some time validating the data passed to your script. Issues such as user error and malicious scripts affect dealing with forms, so it is important you maintain the integrity of whatever device you are using to store information garnered from users.

## WHAT IS FORM?

- ❖ When you login into a website or into your mail box, you are interacting with a form.
- ❖ Forms are used to get input from the user and submit it to the web server for processing.
- ❖ The diagram below illustrates the form handling process.



---

# PHP PROGRAMMING

---

## PHP FORM

- ❖ A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.
- ❖ The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as **input**.

## WHEN AND WHY WE ARE USING FORMS?

- ❖ Forms come in handy when developing flexible and dynamic applications that accept user input.
- ❖ Forms can be used to edit already existing data from the database

## CREATE A FORM

We will use HTML tags to create a form. The list of things used to create a form is

- Opening and closing form tags <form>...</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons, checkboxes etc.
- The code below creates a simple registration form

```
<html>
<head><title>Registration Form</title></head>
<body>
  <form action="registration_form.php" method="POST">
    First name:    <input type="text" name="firstname"> <br>
    Last name:    <input type="text" name="lastname">
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

Viewing the above code in a web browser displays the following form.

# PHP PROGRAMMING

---

## Registration Form



Here,

- `<form...>...</form>` are the opening and closing form tags
- `action="registration_form.php" method="POST">` specifies the destination URL and the submission type.
- First/Last name: are labels for the input boxes
- `<input type="text" ...>` are input box tags
- `<br>` is the new line tag
- `<input type="submit" value="Submit">` is the button that when clicked submits the form to the server for processing

### UNDERSTANDING COMMON FORM ISSUES

When dealing with forms, the most important aspect to remember is that you are limited to a certain variety of fields that can be applied to a form. The fields that have been created are non-negotiable and work in only the way they were created to work. It is important, therefore, to fully understand what is available and how best to use the form features to your advantage.

Element	Description
TEXT INPUT	A simple text box
PASSWORD INPUT	A text box that hides the characters inputted
HIDDEN INPUT	A field that does not show on the form but can contain data
SELECT	A drop-down box with options
LIST	A select box that can have multiple options selected

## PHP PROGRAMMING

---

CHECKBOX	A box that can be checked
RADIO	A radio button that can act as a choice
TEXTAREA	A larger box that can contain paragraph-style entries
FILE	An element that allows you to browse your computer for a file
SUBMIT	A button that will submit the form
RESET	A button that will reset the form to its original state

### SUBMITTING THE FORM DATA TO THE SERVER

When dealing with forms, you must specify the way that the information entered into the form is transmitted to its destination (method=""). The two ways available to a web developer are GET and POST. PHP POST method

**POST:** When Sending data using the POST method is quite a bit more secure (because the method cannot be altered by appending information to the address bar) and can contain as much information as you choose to send.

#### Example:

The example below displays a simple HTML form with two input fields and a submit button:

```
<html>
  <body>
    <form action="welcome.php" method="post">
      Name: <input type="text" name="name"><br>
      E-mail: <input type="text" name="email"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

## PHP PROGRAMMING

---

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables.

### **WELCOME.PHP:**

```
<html>
  <body>
    Welcome <?php echo $_POST["name"]; ?><br>
    Your email address is: <?php echo $_POST["email"]; ?>
  </body>
</html>
```

**Output:**      Welcome Uday  
                 Your email address is udaycse@gmail.com

**GET:** When sending data using the GET method, all fields are appended to the Uniform Resource Locator (URL) of the browser and sent along with the address as data. Sending data using the GET method means that fields are generally capped at 150 characters, which is certainly not the most effective means of passing information. It is also not a secure means of passing data, because many people know how to send information to a script using an address bar.

### **Example**

```
<html>
  <body>
    <form action="welcome_get.php" method="get">
      Name: <input type="text" name="name"><br>
      E-mail: <input type="text" name="email"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

### **WELCOME\_GET.PHP:**

# PHP PROGRAMMING

---

```
<html>
  <body>
    Welcome <?php echo $_GET["name"]; ?><br>
    Your email address is: <?php echo $_GET["email"]; ?>
  </body>
</html>
```

## GET VS POST METHODS

### POST

- ❖ Values not visible in the URL.
- ❖ Has not limitation of the length of the values since they are submitted via the body of HTTP.
- ❖ Has lower performance compared to PHP\_GET method due to time spent encapsulation the PHP\_POST values in the HTTP body
- ❖ Supports many different data types such as string, numeric, binary etc.
- ❖ Results cannot be book marked

### FORM SUBMISSION POST METHOD

```
<form action="registration_form.php" method="POST">
  First name: <input type="text" name="firstname"><br>
  Last name: <input type="text" name="lastname">
  <br>
  <input type="hidden" name="form_submitted" value="1"/>
  <input type="submit" value="Submit">
</form>
```

*Submission URL does not show form values*



localhost/tuttis/registration\_form.php

### GET

- ❖ Values visible in the URL
- ❖ Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
- ❖ Has high performance compared to POST method dues to the simple nature of appending the values in the URL.

---

## PHP PROGRAMMING

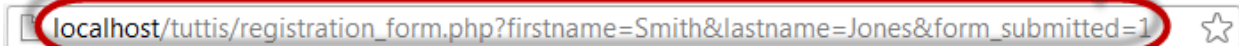
---

- ❖ Supports only string data types because the values are displayed in the URL
- ❖ Results can be book marked due to the visibility of the values in the URL

### FORM SUBMISSION GET METHOD

```
<form action="registration_form.php" method="GET">
  First name: <input type="text" name="firstname"><br>
  Last name: <input type="text" name="lastname">
  <br>
  <input type="hidden" name="form_submitted" value="1"/>
  <input type="submit" value="Submit">
</form>
```

### SUBMISSION URL SHOWS FORM VALUES



localhost/tuttis/registration\_form.php?firstname=Smith&lastname=Jones&form\_submitted=1

### VALIDATING FORM INPUT:

Validation means check whether the field is filled or not in the proper way.

There are two types of validation are available in PHP.

**Client-Side Validation:** Validation is performed on the client machine web browsers.

**Server Side Validation:** Validation is performed on the server machine.

### Some of validation rules for field:

Field	Validation Rules
Name	Should required letters and white-spaces
Email	Should required @ and .
Website	Should required a valid URL



---

## PHP PROGRAMMING

---

Radio	Must be selectable at least once
Check Box	Must be checkable at least once
Drop Down menu	Must be selectable at least once

### EXAMPLES:

**Valid URL:** Below code shows validation of URL

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

Above syntax will verify whether a given URL is valid or not. It should allow some keywords as https, ftp, www, a-z, 0-9,...etc..

**Valid Email:** Below code shows validation of Email address

```
$email = input($_POST["email"]);  
if (!filter_var ($email, FILTER_VALIDATE_EMAIL))  
{  
    $emailErr = "Invalid format and please re-enter valid email";  
}
```

Above syntax will verify whether given Email address is well-formed or not. If it is not, it will show an error message.

### Example Program (Username, Password and Email validation)

```
<html>  
<body>  
<form action="validate.php" method="post">  
    username:<input type="text" name="username">  
    password:<input type="text" name="password" >  
    E-mail:<input type="text" name="email" >  
    <input class="submit" name="submit" type="submit" value="Submit">  
</form>  
</body>
```

## PHP PROGRAMMING

---

</html>

### VALIDATE.PHP

```
<?php
    if(isset($_POST['submit']))
    {
        if (empty($_POST["username"]))
            echo "Name is required."<br>";
        else
        {
            $name = $_POST["username"];
            if (!preg_match("/^[a-zA-Z ]*$/",$name))
                echo "Only letters and white space allowed."<br>";
        }
        if (empty($_POST["password"]))
            echo "Password is required."<br>";
        else
        {
            $pass = $_POST["password"];
            if (!preg_match("/^[a-zA-Z ]*$/",$pass))
                echo "Only letters and white space allowed."<br>";
        }
        if (empty($_POST["email"]))
            echo "Email is required."<br>";
        else
        {
            $email = $_POST["email"];
            if (!preg_match("/^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/",$email))
                echo "Invalid email format."<br>";
                or
            if (!filter_var($email, FILTER_VALIDATE_EMAIL))
                echo "Invalid email format."<br>";
        }
    }
?>
```

### Example program2:

<html>

<body>

<?php

    // define variables and set to empty values

## PHP PROGRAMMING

---

```
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    if (empty($_POST["name"]))
        $nameErr = "Name is required";
    else
        $name = test_input($_POST["name"]);
    if (empty($_POST["email"]))
        $emailErr = "Email is required";
    else
        $email = test_input($_POST["email"]);
    if (!filter_var($email, FILTER_VALIDATE_EMAIL))
        $emailErr = "Invalid email format";
    if (empty($_POST["website"]))
        $website = "";
    else
        $website = test_input($_POST["website"]);
    if (empty($_POST["comment"]))
        $comment = "";
    else
        $comment = test_input($_POST["comment"]);
    if (empty($_POST["gender"]))
        $genderErr = "Gender is required";
    else
        $gender = test_input($_POST["gender"]);
}
function test_input($data)
```

## PHP PROGRAMMING

---

```
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<form method = "post" action = "<?php      echo htmlspecialchars
                                   ($_SERVER["PHP_SELF"]); ?>">

    Name : <input type = "text" name = "name"> E-
    mail: <input type = "text" name = "email">
    Time:  <input type = "text" name = "website">
    Classes: <textarea name="comment"rows="5"cols= "40"></textarea>
    Gender:
        <input type = "radio" name = "gender" value = "female">Female
        <input type = "radio" name = "gender" value = "male">Male
        <input type = "submit" name = "submit" value = "Submit">
</form>
<?php
    echo "<h2>Your given values are as:</h2>";
    echo $name;
    echo "<br>";
    echo $email;
    echo "<br>";
    echo $website;
    echo "<br>";
    echo $comment;
    echo "<br>";
```

## PHP PROGRAMMING

---

```
        echo $gender;
    ?>
</body>
</html>
```

**WORKING WITH MULTIPAGE FORMS:** Sometimes you will need to collect values from more than one page. Most developers do this for the sake of clarity. By providing forms on more than one page, you can separate blocks of information and thus create an ergonomic experience for the user. The problem, therefore, is how to get values from each page onto the next page and finally to the processing script. Being the great developer that you are, you can solve this problem and use the **hidden input** form type. When each page loads, you must load the values from the previous pages into hidden form elements and submit them

### PAGE1.HTML

```
<html >
<head><title> Page1</title></head>
<body>
<form action=" page2.php" method="post">
    Your Name: <input type="text" name="name" maxlength="150" /><br />
    <input type="submit" value="Submit" />
</form>
</body>
</html>
```

### PAGE2.HTML

```
<html>
<head><title> Page2</title></head>
<body>
<form action="page3.php" method="post">
Selection:  <select name="selection">
                <option value="nogo">make a selection...</option>
                <option value="1">Choice 1</option>
                <option value="2">Choice 2</option>
                <option value="3">Choice 3</option>
            </select><br /><br />
<input type="hidden" name="name" value="<?php echo $_POST['name']; ?>" />
<input type="submit" value="Submit" />
```

---

## PHP PROGRAMMING

---

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

### PAGE3.HTML

```
<html>
<head><title> Page3</title></head>
<body>
<form action="page4.php" method="post">
Your Email: <input type="text" name="email" maxlength="150" /><br />
<input type="hidden" name="name" value="<?php echo $_POST['name']; ?>" />
<input type="hidden" name="selection" value="<?php echo $_POST['selection']; ?>"
/>
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

### PAGE4.PHP

```
<html>
<head><title>Page4</title></head>
<body>
<?php
echo "Your Name: " . $_POST['name'] . "<br />";
echo "Your Selection: " . $_POST['selection'] . "<br />";
echo "Your Email: " . $_POST['email'] . "<br />";
?>
<a href="page1.php">Try Again</a>
</body>
</html>
```

**PREVENTING MULTIPLE SUBMISSIONS OF A FORM:** One possible occurrence that happens often is that users become impatient when waiting for your script to do what it is doing, and hence they click the submit button on a form repeatedly. This can cause great damage on your script because, while the user may not see anything happening, your script is probably going ahead with whatever it has been programmed to do. If a user continually hits the submit button on a credit card submittal form, their card may be charged multiple times if the developer has not taken the time to validate against such an eventuality. You can deal with multiple submittal validation in two ways.

---

## PHP PROGRAMMING

---

**PREVENTING MULTIPLE SUBMISSIONS ON THE SERVER SIDE:** we prefer to use a session-based method. Basically, once the submit button has been clicked, the server logs the request from the individual user. If the user attempts to resubmit a request, the script notes a request is already in motion from this user and denies the subsequent request. Once the script has finished processing, the session is unset, and you have no more worries.

```
<html>
<body>
<form action="process.php" method="post">
Your Name:<input type="text" name="name" ><br />
<input type="submit" value="Submit" >
</form>
</body>
</html>
```

### PROCESS.PHP

```
<?php
    session_start ();
    if (!isset ($_SESSION['processing']))
        $_SESSION['processing'] = false;
    if ($_SESSION['processing'] == false)
    {
        $_SESSION['processing'] = true;
        for ($i = 0; $i < 2000000; $i++)
        {
            //Thinking...
        }
        if ($file = fopen ("test.txt","w+"))
            fwrite ($file, "Processing");
        else
            echo "Error opening file.";
        echo $_POST['yourname'];
        unset ($_SESSION['processing']);
    }
?>
```

**PREVENTING MULTIPLE SUBMISSIONS ON THE CLIENT SIDE:** Handling multiple submittals from a client-side perspective is actually much simpler than doing it on the server side. With well-placed JavaScript, you can ensure that the browser will not let the submittal go through more than once.

## PHP PROGRAMMING

---

```
<html>
<head>
<script language="javascript" type="text/javascript">
    function checkandsubmit()
    {        //Disable the submit button.
            document.test.submitbut.disabled = true;
            //Then submit the form.
            document.test.submit();
    }
</script>
</head>
<body>
<form        action="process.php"method="post"name="test"onsubmit="return
checkandsubmit ()">
Your Name: <input type="text" name="name" maxlength="150" /><br />
<input      type="submit"value="Submit"id="submitbut"        name="submitbut"/>
</form>
</body>
</html>
```

### PROCESS.PHP

```
<?php
for ($i = 0; $i < 2000000; $i++)
{
    //Thinking...
}
if ($file = fopen ("test.txt","w+"))
    fwrite ($file, "Processing");
else
    echo "Error opening file.";
echo $_POST['yourname'];
?>
```

### PROCESSING THE REGISTRATION FORM DATA

- ❖ The registration form submits data to itself as specified in the action attribute of the form.
- ❖ When a form has been submitted, the values are populated in the \$\_POST super global array.
- ❖ We will use the PHP isset function to check if the form values have been filled in the \$\_POST array and process the data.



## PHP PROGRAMMING

---

- ❖ We will modify the registration form to include the PHP code that processes the data. Below is the modified code

```
<html>
<head><title>Registration Form</title></head>
<body>
    <?php if (isset($_POST['form_submitted'])): ?>
//this code is executed when the form is submitted
    <h2>Thank You <?php echo $_POST['firstname']; ?> </h2>
    <p>You have been registered as
        <?php echo $_POST['firstname'] . ' ' . $_POST['lastname']; ?>
    </p>
    <p>Go <a href="/registration_form.php">back</a> to the form</p>
    <?php else: ?>
        <h2>Registration Form</h2>
        <form action="registration_form.php" method="POST">
            First name:
            <input type="text" name="firstname">
            <br> Last name:
            <input type="text" name="lastname">
                <input type="hidden" name="form_submitted" value="1" />
            <input type="submit" value="Submit">
        </form>
    <?php endif; ?>
</body>
</html>
```

Here,

## PHP PROGRAMMING

---

- `<?php if (isset($_POST['form_submitted'])): ?>` checks if the `form_submitted` hidden field has been filled in the `$_POST[]` array and display a thank you and first name message.
- If the `form_submitted` field hasn't been filled in the `$_POST[]` array, the form is displayed.

### PHP & MYSQL

#### What is MySQL?

MySQL is open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL), which is used for adding ,accessing and managing content in database.It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation.

- The data in a MySQL database are stored in a table which consists of columns and rows.
- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable and easy to use database system.
- MySQL compiles on a number of platforms.

**PHPMYADMIN:** **PHPMYADMIN** is a open source platform used for administering MySQL with a web browser. By using **PHPMYADMIN** we can perform database operations such as create, alter, drop, delete, import and export database tables. **PHPMYADMIN** is available with XAMPP installation. For accessing **PHPMYADMIN** we can use the following URL address

**<https://localhost/phpmyadmin>**

❖ It is providing number of options to manage MySQL database tables.

**Insert:** Using this option, we can insert a record or set of records.

**Browse:** Using this option, we can display all records of tables.

---

## PHP PROGRAMMING

---

**Structure:** To change the structure of table. If you want to add a column or delete existing columns, to add a constraint or delete existing constraints we can use this option.

**SQL:** To execute SQL queries.

**Search:** Using this option, we can search tables records in ascending or descending order based on specified column.

**Operations:** To change the options of table like table name, storage engine, etc.

**Export:** Using this option, we can export table data as SQL file, pdf file, xml file, etc.

**Import:** Using this option, we can import exported SQL file into current database.

**Empty:** Using this option, we can delete all records of table.

**Drop:** Using this option, we can drop structure of table.

### DATATYPES

In MySQL 3 types of data types available. They are

- a. Numeric data types
- b. String data types
- c. Date and Time data types
- a. Numeric data types

This data type can store numeric values. It is divided into 2 types.

- 1. Signed data type
- 2. Unsigned data type

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215

## PHP PROGRAMMING

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	$-2^{63}$	0	$2^{63}-1$	$2^{64}-1$

- **FLOAT (10,2)**

This data type can store decimal values. We can store total digits 10 and after decimal 2 digits.

- **DOUBLE (16,4)**

We can store total number of digits are 16 and after decimal 4.

- **DECIMAL**

Same as floating point number but cannot be unsigned.

### b. String data types

String Types	Description
<a href="#">CHAR</a>	A fixed-length nonbinary (character) string
<a href="#">VARCHAR</a>	A variable-length non-binary string
BINARY	A fixed-length binary string
VARBINARY	A variable-length binary string
TINYBLOB	A very small BLOB (binary large object)
BLOB	A small BLOB
MEDIUMBLOB	A medium-sized BLOB
LOB	A large BLOB
<a href="#">TINYTEXT</a>	A very small non-binary string
<a href="#">TEXT</a>	A small non-binary string
<a href="#">MEDIUMTEXT</a>	A medium-sized non-binary string
<a href="#">LONGTEXT</a>	A large non-binary string

## PHP PROGRAMMING

---

String Types	Description
<a href="#">ENUM</a>	An enumeration; each column value may be assigned one enumeration member
SET	A set; each column value may be assigned zero or more SET members

### c. Date and Time data types

Date and Time Types	Description
<a href="#">DATE</a>	A date value in YYYY-MM-DD format
<a href="#">TIME</a>	A time value in hh:mm:ss format
<a href="#">DATETIME</a>	A date and time value in YYYY-MM-DD hh:mm:ss format
<a href="#">TIMESTAMP</a>	A timestamp value in YYYY-MM-DD hh:mm:ss format
YEAR	A year value in YYYY or YY format

### **ESTABLISH A CONNECTION WITH DATABASE:**

**mysqli\_connect():** Using this function, we can establish a connection with MySQL database. Arguments are server name, user id and password.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$conn = mysqli_connect($servername, $username, $password);
```

```
if (!$conn)
    die("Connection failed: " . mysqli_connect_error());
echo "Connected successfully";
?>
```

**Or**

```
<?php
$con=mysqli_connect("localhost","root","");
```

## PHP PROGRAMMING

---

```
if($con)
    echo "connected";
else
    echo "Not connected";

?>
```

### **Example: To create a database in MySQL**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$conn = mysqli_connect($servername, $username, $password);
if (!$conn)
    die("Connection failed: " . mysqli_connect_error());
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql))
    echo "Database created successfully";
else
    echo "Error creating database: " . mysqli_error($conn);
mysqli_close($conn);
?>
```

### **Example: To select a database in MySQL**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$conn = mysqli_connect($servername, $username, $password);
if (!$conn)
    die("Connection failed: " . mysqli_connect_error());
```

## PHP PROGRAMMING

---

```
$sql =mysql_select_db($conn,"myDB");
if (mysqli_query($conn, $sql))
    echo "Database selected successfully";
else
    echo "Error selecting database: " . mysqli_error($conn);
mysqli_close($conn);
?>
```

### **Example: To create a table in MySQL**

```
<?php
$link = mysqli_connect("localhost", "root", "");
if($link == false)
    die("ERROR: Could not connect. " . mysqli_connect_error());
$sql = "CREATE TABLE persons(
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    email VARCHAR(70) NOT NULL UNIQUE )";
if(mysqli_query($link, $sql))
    echo "Table created successfully.";
else
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
mysqli_close($link);
?>
```

### **Example: To insert a record in a table**

```
<?php
$link = mysqli_connect("localhost", "root", "", "");
if($link == false)
    die("ERROR: Could not connect. " . mysqli_connect_error());
```

## PHP PROGRAMMING

---

```
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES ('Peter',  
'Parker', 'peterparker@mail.com')";  
if(mysqli_query($link, $sql))  
    echo "Records inserted successfully.";  
else  
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);  
mysqli_close($link);  
?>
```

- ❖ **mysqli\_error()**: To get error message if any occurred at the time of MySQL statement execution.
- ❖ **mysqli\_errno()**: To get error number if any occurred at the time of SQL statement execution.

```
<?php  
$con=mysqli_connect("localhost","root","");  
if (!$con)  
{  
    die("Connection error: " . mysqli_connect_errno());  
}  
?>
```

### Selecting Data From Database Tables

The SQL SELECT statement is used to select the records from database tables.

```
<?php  
$link = mysqli_connect("localhost", "root", "");  
if($link == false)  
    die("ERROR: Could not connect. " . mysqli_connect_error());  
$sql = "SELECT * FROM persons";  
if($result = mysqli_query($link, $sql))  
{
```



## PHP PROGRAMMING

---

```
if(mysqli_num_rows($result) > 0)
{
    echo "<table>";
    echo "<tr>";
        echo "<th>id</th>";
        echo "<th>first_name</th>";
        echo "<th>last_name</th>";
        echo "<th>email</th>";
    echo "</tr>";
    while($row = mysqli_fetch_array($result))
    {
        echo "<tr>";
            echo "<td>" . $row['id'] . "</td>";
            echo "<td>" . $row['first_name'] . "</td>";
            echo "<td>" . $row['last_name'] . "</td>";
            echo "<td>" . $row['email'] . "</td>";
        echo "</tr>";
    }
    echo "</table>";
    // Free result set
    mysqli_free_result($result);
}
else
{
    echo "No records matching your query were found.";
}
}
```

## PHP PROGRAMMING

---

```
else
{
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
mysqli_close($link);
?>
```

### Filtering the Records

The WHERE clause is used to extract only those records that fulfill a specified condition.

```
<?php
$link = mysqli_connect("localhost", "root", "");
if($link === false)
    die("ERROR: Could not connect. " . mysqli_connect_error());
$sql = "SELECT * FROM persons WHERE first_name='john'";
if($result = mysqli_query($link, $sql))
{
    if(mysqli_num_rows($result) > 0)
    {
        echo "<table>";
        echo "<tr>";
        echo "<th>id</th>";
        echo "<th>first_name</th>";
        echo "<th>last_name</th>";
        echo "<th>email</th>";
        echo "</tr>";
        while($row = mysqli_fetch_array($result))
        {
            echo "<tr>";
```

## PHP PROGRAMMING

---

```
        echo "<td>" . $row['id'] . "</td>";
        echo "<td>" . $row['first_name'] . "</td>";
        echo "<td>" . $row['last_name'] . "</td>";
        echo "<td>" . $row['email'] . "</td>";
    echo "</tr>";
}
echo "</table>";
// Close result set
mysqli_free_result($result);
}
else
{
    echo "No records matching your query were found.";
}
}
else
{
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
mysqli_close($link);
?>
```

### Ordering the Result Set

The ORDER BY clause can be used in conjugation with the SELECT statement to see the data from a table ordered by a specific field. The ORDER BY clause lets you define the field name to sort against and the sort direction either ascending or descending.

```
<?php
```

```
$link = mysqli_connect("localhost", "root", "");
```

## PHP PROGRAMMING

---

```
if($link === false)
    die("ERROR: Could not connect. " . mysqli_connect_error());
$sql = "SELECT * FROM persons ORDER BY first_name";
if($result = mysqli_query($link, $sql))
{
    if(mysqli_num_rows($result) > 0)
    {
        echo "<table>";
        echo "<tr>";
            echo "<th>id</th>";
            echo "<th>first_name</th>";
            echo "<th>last_name</th>";
            echo "<th>email</th>";
        echo "</tr>";
        while($row = mysqli_fetch_array($result))
        {
            echo "<tr>";
                echo "<td>" . $row['id'] . "</td>";
                echo "<td>" . $row['first_name'] . "</td>";
                echo "<td>" . $row['last_name'] . "</td>";
                echo "<td>" . $row['email'] . "</td>";
            echo "</tr>";
        }
        echo "</table>";
    }
    else
    {
        echo "No records matching your query were found.";
    }
}
```

## PHP PROGRAMMING

---

```
    }  
}  
else  
{  
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);  
}  
mysqli_close($link);  
?>
```

### Updating Database Table Data

The UPDATE statement is used to change or modify the existing records in a database table. This statement is typically used in conjugation with the WHERE clause to apply the changes to only those records that matches specific criteria.

```
<?php  
$link = mysqli_connect("localhost", "root", "");  
if($link === false)  
    die("ERROR: Could not connect. " . mysqli_connect_error());  
$sql = "UPDATE persons SET email='peterparker_new@mail.com' WHERE  
id=1";  
if(mysqli_query($link, $sql))  
    echo "Records were updated successfully.";  
else  
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);  
mysqli_close($link);  
?>
```

### Deleting Database Table Data

Just as you insert records into tables, you can delete records from a table using the SQL DELETE statement. It is typically used in conjugation with the

## PHP PROGRAMMING

---

WHERE clause to delete only those records that matches specific criteria or condition.

```
<?php
$link = mysqli_connect("localhost", "root", "");
if($link === false)
    die("ERROR: Could not connect. " . mysqli_connect_error());
$sql = "DELETE FROM persons WHERE first_name='John'";
if(mysqli_query($link, $sql))
    echo "Records were deleted successfully.";
else
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
mysqli_close($link);
?>
```

# PHP PROGRAMMING

---

## **SUPER GLOBAL VARIABLES**

- ❖ PHP super global variables are used to access global variables from anywhere in the PHP script.
- ❖ PHP Super global variables are accessible inside the same page that defines it, as well as outside the page.
- ❖ While local variable's scope is within the page that defines it.
- ❖ These are specially-defined array variables in PHP that make it easy for you to get information about a request or its context. The super global are available throughout your script.
- ❖ These variables can be accessed from any function, class or any file without doing any special task such as declaring any global variable etc. They are mainly used to store and get information from one page to another in an application.
- ❖ The list of super global variables available in PHP are:
  - \$GLOBALS

## PHP PROGRAMMING

---

- \$\_SERVER
- \$\_REQUEST
- \$\_GET
- \$\_POST
- \$\_SESSION
- \$\_COOKIE
- \$\_FILES
- \$\_ENV

### 1. \$GLOBALS

- ❖ It is a super global variable which is used to access global variables from anywhere in the PHP script.
- ❖ PHP stores all the global variables in array \$GLOBALS [ ] it consist of an array which have an index that holds the global variable name, which can be accessed.

#### Example:

```
<?php
    $x = 300;
    $y = 200;
    function multiplication()
    {
        $GLOBALS['z'] = $GLOBALS['x'] * $GLOBALS['y'];
    }
    multiplication();
    echo $z;
?>
```

#### Output:

60000

### 2. \$\_SERVER



---

## PHP PROGRAMMING

---

- ❖ It is a PHP super global variable that stores the information about headers, paths and script locations (i.e., complete information of web server and web browser).
- ❖ Some of these elements are used to get the information from the super global variable \$\_SERVER.

**Example:**

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

**Output:**

All server related variables will be shown

The following table lists the most important elements that can go inside \$\_SERVER:

Element/Code	Description
\$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
\$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
\$_SERVER['SERVER_ADDR']	Returns the IP address of the host server

## PHP PROGRAMMING

<code>\$_SERVER['SERVER_NAME']</code>	Returns the name of the host server (such as www.w3schools.com)
<code>\$_SERVER['SERVER_SOFTWARE']</code>	Returns the server identification string (such as Apache/2.2.24)
<code>\$_SERVER['SERVER_PROTOCOL']</code>	Returns the name and revision of the information protocol (such as HTTP/1.1)
<code>\$_SERVER['REQUEST_METHOD']</code>	Returns the request method used to access the page (such as POST)
<code>\$_SERVER['REQUEST_TIME']</code>	Returns the timestamp of the start of the request (such as 1377687496)
<code>\$_SERVER['QUERY_STRING']</code>	Returns the query string if the page is accessed via a query string
<code>\$_SERVER['HTTP_ACCEPT']</code>	Returns the Accept header from the current request
<code>\$_SERVER['HTTP_ACCEPT_CHARSET']</code>	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)

## PHP PROGRAMMING

<code>\$_SERVER['HTTP_HOST']</code>	Returns the Host header from the current request
<code>\$_SERVER['HTTP_REFERER']</code>	Returns the complete URL of the current page (not reliable because not all user-agents support it)
<code>\$_SERVER['HTTPS']</code>	Is the script queried through a secure HTTP protocol
<code>\$_SERVER['REMOTE_ADDR']</code>	Returns the IP address from where the user is viewing the current page
<code>\$_SERVER['REMOTE_HOST']</code>	Returns the Host name from where the user is viewing the current page
<code>\$_SERVER['REMOTE_PORT']</code>	Returns the port being used on the user's machine to communicate with the web server
<code>\$_SERVER['SCRIPT_FILENAME']</code>	Returns the absolute pathname of the currently executing script
<code>\$_SERVER['SERVER_ADMIN']</code>	Returns the value given to the <code>SERVER_ADMIN</code> directive in the web server configuration file (if your

---

## PHP PROGRAMMING

---

	script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
<code>\$_SERVER['SERVER_PORT']</code>	Returns the port on the server machine being used by the web server for communication (such as 80)
<code>\$_SERVER['SERVER_SIGNATURE']</code>	Returns the server version and virtual host name which are added to server-generated pages
<code>\$_SERVER['PATH_TRANSLATED']</code>	Returns the file system based path to the current script
<code>\$_SERVER['SCRIPT_NAME']</code>	Returns the path of the current script
<code>\$_SERVER['SCRIPT_URI']</code>	Returns the URI of the current page

### 3. `$_REQUEST`

- ❖ The PHP `$_REQUEST` variable contains the contents of both `$_GET`, `$_POST`, and `$_COOKIE`.
- ❖ The PHP `$_REQUEST` variable can be used to get the result from form data sent with both the GET and POST methods. By using this we can also get values of cookie and query string.

---

## PHP PROGRAMMING

---

- ❖ Query string is small amount of data on URL address followed by ? symbol.
- ❖ Along with form data, if we want to transfer some extra information from one page to another page at the time of redirection, we can use query string.
- ❖ Query string can be passed in two ways:
  1. Query string as name and value.
  2. Query string with only value.
- ❖ Using `$_REQUEST`, we can get value of query string if it is combination of name and value.

### Example:

#### Abc.html

```
<a href="p1.php? sno=100 & un="scott">click</a>
```

#### p1.php

```
<?php
    echo $_REQUEST["sno"];
    echo $_REQUEST["un"];
?>
```

### 4. `$_GET`

- ❖ It is an array data type. Total number of elements is equal to total number of posted values.
- ❖ Elements keys are control names and element values are control values.
- ❖ The `$_GET` variable is used to collect values from a form with `method="get"`.
- ❖ Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 100 characters).

### Example:

## PHP PROGRAMMING

---

### Form.html

```
<form method="GET" action="p1.php">
    <input type="text" name="t1" ><br>
    <input type="text" name="t2" ><br>
    <input type="submit" name="b1" value="click" ><br>
</form>
```

### P1.php

```
<?php
    print_r($_GET);
    echo $_GET["t1"];
    echo $_GET["t2"];
    echo $_GET["b1"];
?>
```

### Output:

Array([t1]->uday [t2]->cse [sub]->click)

uday cse click

### 5. \$\_POST

- ❖ The \$\_POST variable is an array of variable names and values sent by the HTTP POST method.
- ❖ The \$\_POST variable is used to collect values from a form with method="post".
- ❖ Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- ❖ If more than one submit controls are there in the form, then only one control should pass to server.

### Example:

### Form.php

```
<?php
```

## PHP PROGRAMMING

---

```
$txt1=$_POST['t1'];
$txt2=$_POST['t2'];
echo "value is".$txt1;
echo "value is".$txt2;
?>

<form method="GET" action="" ">
    <input type="textbox" name="t1" ><br>
    <input type="textbox" name="t2" ><br>
    <input type="submit" name="b1" value="click" ><br>
</form>
```

**NOTE:** In the above example, the php script is first executed. Since the form is not yet executed, the values are undefined for txt1 and txt2 variables. Thus to eliminate this problem, we rewrite the code as:

```
<?php
    if(isset($_POST['b1']))
    {
        error_reporting(E_ALL);
        $txt1=$_POST['t1'];
        $txt2=$_POST['t2'];
        echo "value is".$txt1;
        echo "value is".$txt2;
    }
?>

<form method="GET" action="" ">
    <input type="textbox" name="t1" ><br>
    <input type="textbox" name="t2" ><br>
    <input type="submit" name="b1" value="click" ><br>
</form>
```

---

## PHP PROGRAMMING

---

### How to post the values in the textbox after reloading of form?

```
<?php
    if(isset($_POST['b1']))
    {
        error_reporting(E_ALL);
        $txt1=$_POST['t1'];
        $txt2=$_POST['t2'];
        echo "value is".$txt1;
        echo "value is".$txt2;
    }
?>

<form method="GET" action=" ">
    <input type="textbox" name="t1" value="<?php echo $_POST['t1']; ?>"><br>
    <input type="textbox" name="t2" value="<?php echo $_POST['t2']; ?>"><br>
    <input type="submit" name="b1" value="click" ><br>
</form>
```

### 6. \$\_SESSION

- ❖ A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.
- ❖ It is same as cookies used to maintain state of application.
- ❖ Data of session storing in server memory location can be accessed from any webpage of that application.
- ❖ Using this variable we can create and access sessions.
- ❖ By default, we cannot access sessions of one page from another page. To access sessions, we need to initialize them in global memory locations.
- ❖ There are two ways to initialize sessions in global memory locations:



## PHP PROGRAMMING

---

- A. Change session auto\_start configuration setting value as 1. This is used to initialize sessions.
- B. Use session\_start() in the program from where you want to access sessions.

### SESSION ID:

- ❖ Session Id is an alphanumeric string generated by webserver when user sends a request to server.
- ❖ If user sends a request, server checks the request whether it contains session id or not. If request doesn't contain session id then server creates a new session id.
- ❖ It is alphanumeric string. At the same time in server temporary memory location, a file will be created to maintain the session data of user. File name is same as session id with prefix word **sess\_**.
- ❖ Server sends a response to client system along with session id and storing that session id as in-memory cookie. The name of cookie is **PHPSESSID** and the value is session id.
- ❖ From next request onwards same session id transfers between browser and server.
- ❖ Server is using session id to identify users. If user close browser in-memory cookie will be destroyed. Later, if user sends request to the server, server sends the request without session cookie. Again browser will create new session id.
  - i. **session\_id()**
  - ii. **session\_unregister()**
  - iii. **session\_unset()**
  - iv. **session\_destroy()**
  - v.

### PHP Session Variables

## PHP PROGRAMMING

---

- ❖ When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end.
- ❖ But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.
- ❖ A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.
- ❖ Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

### Starting a PHP Session

- ❖ Before you can store user information in your PHP session, you must first start up the session.
- ❖ The `session_start()` function must appear BEFORE the `<html>` tag:

```
<?php session_start(); ?>
<html>
    <body>
        </body>
</html>
```

- ❖ The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

### Storing a Session Variable

## PHP PROGRAMMING

---

- ❖ The correct way to store and retrieve session variables is to use the PHP `$_SESSION` variable:

### Example:

```
<?php
    session_start();
    // store session data
    $_SESSION['views']=1;
?>
<html>
    <body>
        <?php
            //retrieve session data
            echo "Pageviews=". $_SESSION['views'];
        ?>
    </body>
</html>
```

### Output:

Pageviews=1

In the example below, we create a simple page-views counter. The `isset()` function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
    session_start();
    if(isset($_SESSION['views']))
        $_SESSION['views']=$_SESSION['views']+1;
    else
```

## PHP PROGRAMMING

---

```
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

### Destroying a Session

- ❖ If you wish to delete some session data, you can use the unset() or the session\_destroy() function.
- ❖ The unset() function is used to free the specified session variable:
- ❖ The session\_destroy() will reset your session and you will lose all your stored session data.

```
<?php
unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session\_destroy() function:

```
<?php
session_destroy();
?>
```

### 7. \$\_COOKIE

- ❖ Cookie is a state management object used to maintain state of application.
- ❖ Data of cookie will store in browser memory location. This data can be accessed from any web page in an application.
- ❖ A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.
- ❖ Cookie is state
- ❖ A cookie is often used to identify a user.

# PHP PROGRAMMING

---

❖ Cookies are divided into two types:

## 1. In-memory cookie

We create a cookie in client machine without explicit expiry time is called in-memory cookie. Data of this cookie will delete after closing web browser.

## 2. Persistence cookie

We create a cookie in client machine with explicit expiry time is called persistence cookie. It stores data in hard disk memory location, that data will destroy after its lifetime is completed.

- ❖ Every browser contains memory locations in RAM and hard disks. Hard disk memory allocates when the browser is installed. Ram memory allocates when we open the first window in browser.
- ❖ RAM memory deallocates when we close all windows of browser. Hard disk memory deallocates after uninstallation of browser.
- ❖ Memory of one browser cannot be accessible by another browser.
- ❖ All windows of in browser can share common memory allocation.
- ❖ Every website contains a folder in browser. Cookies of that website will store in that folder only. This is the reason cookies of one website cannot be accessed by the cookies of another website. This folder will be created when website is opening in browser.

## HOW TO CREATE A COOKIE?

- ❖ The `setcookie()` function is used to create a cookie in client machine.
- ❖ It must appear BEFORE the `<html>` tag.
- ❖ The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use `setrawcookie()` instead).

## Syntax

**`setcookie(name, value, expire, path, domain);`**

## Example:

## PHP PROGRAMMING

---

```
<?php
    setcookie("user", "Alex Porter", time()+3600);
?>
```

### Example

```
<?php
    $expire=time()+60*60*24*30;
    setcookie("user", "Alex Porter", $expire);
?>
```

### Example

#### Pro.php

```
<?php
    setcookie(x,100);
    echo $_COOKIE['x'];
?>
<a href="p1.php">GO</a>
```

#### P1.php

```
<?php
    echo "HI";
    echo $_COOKIE['x'];
?>
```

### NOTE:

At the time of 1<sup>st</sup> execution, the value of x is not shown in the output. Since the time setcookie() is creating the cookie, the next statement is executed. This setcookie() takes some time to create cookie, thus we are unable to see value of x at first execution.

### HOW TO VIEW COOKIES IN BROWSER?

## PHP PROGRAMMING

---

Open your Browser ☐ Tools ☐ Privacy ☐ Custom selection for history ☐  
☐ Show cookie ☐ Cookies folder will be shown

### HOW TO RETRIEVE A COOKIE VALUE?

❖ The PHP \$\_COOKIE variable is used to retrieve a cookie value.

```
<?php
    // Print a cookie
    echo $_COOKIE["user"];
    // A way to view all cookies
    print_r($_COOKIE);
?>
```

In the following example we use the isset() function to find out if a cookie has been set:

#### Example:

```
<html>
    <body>
        <?php
            if (isset($_COOKIE["user"]))
                echo "Welcome " . $_COOKIE["user"] . "!"<br />";
            else
                echo "Welcome guest!"<br />";
        ?>
    </body>
</html>
```

### HOW TO DELETE A COOKIE?

When deleting a cookie you should assure that the expiration date is in the past.

#### Example:

## PHP PROGRAMMING

---

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

### HOW TO CREATE PERSISTENCE COOKIES?

There are four steps to create persistence cookies.

1. Find out current date and time information when cookie is downloading into client system.
2. Add lifetime to current date and time to get expiry time.
3. Create cookie in client system with the resultant expiry time.

```
<?php
    setcookie("sno",123,time()+3600);

                Current time      Expiry(in Sec)

    echo $_COOKIE['sno'];
?>
```

4. We can destroy the cookies before its expiry time by recreating the cookie with completed time.

```
<?php
    setcookie("sno",'',time()-1);
?>
```

### WHAT IF A BROWSER DOES NOT SUPPORT COOKIES?

- ❖ If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to pass the data through forms (forms and user input are described earlier in this tutorial).
- ❖ The form below passes the user input to "welcome.php" when the user clicks on the "Submit" button:

```
<html>
```



## PHP PROGRAMMING

---

```
<body>
    <form action="welcome.php" method="post">
        Name: <input type="text" name="name" />
        Age: <input type="text" name="age" />
        <input type="submit" />
    </form>
</body>
</html>
```

Retrieve the values in the "welcome.php" file like this:

```
<html>
    <body>
        Welcome <?php echo $_POST["name"]; ?>.<br />
        You are <?php echo $_POST["age"]; ?> years old.
    </body>
</html>
```

### **LIMITATIONS OF COOKIE:**

1. It can store limited amount of data.
2. It can store only text data.
3. Data of cookie is storing in browser memory location, that's why we should store secure information in cookies. If you want to store secure data, we should get user's permission.

### **Program: Pro.php**

```
<?php
    if(isset($_POST['sub']))
    {
        $sp=$_POST['d1'];
```

## PHP PROGRAMMING

---

```
$qty=$_POST['tqty'];
setcookie($sp,$qty);
}
?>
<form method="post" action=" ">
    Select Product:<select name="d1">
        <option>Samsung</option>
        <option>Nokia</option>
        <option>Apple</option>
    </select><br>
    Enter Quantity:<input name="tqty">
    <br>
    <input name="submit" type="submit" value="Add to Cart">
</form>
<a href="bill.php">Bill</a>
```

### **Bill.php**

```
<?php
    foreach($_COOKIE as $k => $v)
    {
        echo $k;
        echo "=";
        echo $v;
        echo "<br>";
    }
?>
```

### **8. \$\_FILES**

- ❖ By using this, we can get information of uploaded file.

## PHP PROGRAMMING

---

- ❖ It is a 2-D array variable that contains 5 elements. Each element is providing information about uploaded files. Every element's first dimension is name of upload control.

### **BASIC CONCEPTS OF UPLOADING FILE:**

- ❖ If user uploaded any file from browser to server, first that file transfers to temporary memory location of server. We need to implement PHP script to move that file from temporary memory location to permanent memory location.
- ❖ Server will provide unique name at the time of storing file in temporary memory location.

### **ELEMENTS OF \$\_FILES:**

- A. **\$\_FILES['file']['tmp\_name']** – To get temporary filename which is provided by server.
- B. **\$\_FILES['file']['name']** – Holds the actual name of the uploaded file.
- C. **\$\_FILES['file']['size']** – Holds the size in bytes of the uploaded file.
- D. **\$\_FILES['file']['type']** – Holds MIME type of uploaded file.
- E. **\$\_FILES['file']['error']** – Error occurred at the time of uploading a file.
  - To get error number, if any occurred at the time of uploading file.
  - If number is 0, there is no error.
  - If number is 1, then file size is maximum than server configuration setting.
  - If number is 2, then file size is maximum than browser configuration setting.

## PHP PROGRAMMING

---

- If number is 3, network problem at the time of uploading a file.
- If number is 4, user select submit button without any file selection.

### Example:

<?php

```
if(isset($_FILES['image']))
{
    $errors= array();
    $file_name = $_FILES['image']['name'];
    $file_size = $_FILES['image']['size'];
    $file_tmp = $_FILES['image']['tmp_name'];
    $file_type = $_FILES['image']['type'];
    $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));
    $extensions= array("jpeg","jpg","png");
    if(in_array($file_ext,$extensions)=== false)
    {
        $errors[]="extension not allowed, please choose a JPEG or PNG
file.";
    }
    if($file_size > 2097152) {
        $errors[]='File size must be excately 2 MB';
    }

    if(empty($errors)==true) {
        move_uploaded_file($file_tmp,"images/".$file_name);
        echo "Success";
    }
    else{
        print_r($errors);
    }
}
```

## PHP PROGRAMMING

---

```
    }  
}  
?>  
<html>  
  <body>  
    <form action = "" method = "POST" enctype = "multipart/form-data">  
      <input type = "file" name = "image" />  
      <input type = "submit"/>  
      <ul>  
        <li>Sent file: <?php echo $_FILES['image']['name']; ?>  
        <li>File size: <?php echo $_FILES['image']['size']; ?>  
        <li>File type: <?php echo $_FILES['image']['type']; ?>  
      </ul>  
    </form>  
  </body>  
</html>
```

### Output:



The screenshot shows a web form with a file upload area. It includes a 'Choose File' button, the text 'No file chosen', and a 'Submit' button. Below the upload area, there is a bulleted list showing the output of the PHP script:

- Sent file:
- File size:
- File type:

### MIME:

- ❖ It stands for **M**ultipurpose **I**nternet **M**ailing **E**xtension.
- ❖ It is a type of extension used to upload a file from browser to server.
- ❖ Every file contains different types of MIME types to upload into server from browser.
- ❖ Available MIME types are:
  1. jpg - image/jpeg

## PHP PROGRAMMING

---

- 2. bmp - image/bmp
- 3. exe - application/octet-stream
- 4. pdf - application/pdf
- 5. multipart/form-data - using this we can upload any type of file from browser to server.
- 6. is\_uploaded\_file - using this function, we can check a file can be uploaded or not from temporary memory location to permanent memory location.
- 7. move\_uploaded\_file - to move the uploaded file from temporary memory location to permanent memory location.

### **Example: select.html**

```
<form method="POST" enctype="multipart/form-data" action="upload.php">  
    Select File:<input name="f1" type="file"> <br>  
    <input name="sub" type="submit" value="upload">  
</form>
```

### **Upload.php**

```
<?php>  
    print_r($_FILES);  
    $fname=$_FILES['f1']['name'];  
    echo $fname;  
    if(move_uploaded_file($_FILES['f1']['tmp_name'], "up/$fname"))  
        echo "File is moved";  
    else  
        echo "Not";  
?>
```

### **Example: To allow all files except .exe files.**

```
<?php  
    if($_FILES['f1']['type']!="application/octet_stream")
```

## PHP PROGRAMMING

---

```
{
    echo "exe not supported"
}
else
    move_uploaded_file($_FILES['f1']['tmp_name'], "up/".$_FILES['f1']['name']);
?>
```

- ❖ Without MIME types, we won't be able to upload file, only txt controls will be supported.

### CONFIGURATION SETTINGS TO WORK WITH FILE UPLOAD CONCEPT:

#### **file\_upload**

- ❖ Using this we can allow and stop file uploads. Default value is ON, by changing this value as OFF we can stop file uploads.

#### **upload\_tmp\_dir**

- ❖ Using this we can change temporary directory location of uploaded file. By default, all files will store in tmp folder.

#### **upload\_max\_filesize**

- ❖ Using this we can increase or decrease maximum file size to upload the files (by default 128MB).

### PROTOCOLS:

- ❖ Protocols are set of instructions to transfer data from one page to another page.
- ❖ Protocols are mainly divided into two types:

#### **1. Stateful protocols:**

These protocols can maintain the state of application. i.e. they can remember previous pages data in current pages. In windows application, we are using these protocols.

E.g., TCP/IP, FTP etc.

#### **2. Stateless protocols:**

## PHP PROGRAMMING

These protocols cannot maintain state of application. In web application, we are using these protocols because they don't carry previous data into current page, that's why performance is very fast.

E.g., HTTP, HTTPS

### 9. \$\_ENV

- ❖ \$\_ENV global variable in PHP is a predefined reserved variable that contains an array of information related to the environment in which PHP script is running.

#### Example:

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

- ❖ By using this function, we can get all configuration settings.

```
<?php
    phpinfo();
?>
```

The World of Fruit

Fruit Survey

Name

Address

Email  Missing

How many pieces of fruit do you eat per day?

☐ 0

☐ 1

☐ 2

☐ More than 2

My favourite fruit

Apple

Banana

Plum

Pomegranate

Would you like a brochure? ☐

Submit

## VALIDATING FORM INPUT



## PHP PROGRAMMING

---

Required field will check whether the field is filled or not in the proper way.

Most of cases we will use the \* symbol for required field.

### What is Validation?

Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows –

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

**Absolute classes registration**  
  

\* required field.

Name:

E-mail:

Time:

Classes:

Gender: ☐ Female ☐ Male

**Your given values are as :**

```
<html>  
  <body>  
    <?php  
      $nameErr = $emailErr = $genderErr = $websiteErr = "";
```

## PHP PROGRAMMING

---

```
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    if (empty($_POST["name"]))
    {
        $nameErr = "Name is required";
    }
    else
    {
        $name = test_input($_POST["name"]);
    }
    if (empty($_POST["email"]))
    {
        $emailErr = "Email is required";
    }
    else
    {
        $email = test_input($_POST["email"]);
        if (!filter_var($email, FILTER_VALIDATE_EMAIL))
        {
            $emailErr = "Invalid email format";
        }
    }
    if (empty($_POST["website"]))
    {
        $website = "";
    }
    else
    {
        $website = test_input($_POST["website"]);
    }
    if (empty($_POST["comment"]))
    {
        $comment = "";
    }
    else
    {
        $comment = test_input($_POST["comment"]);
    }
    if (empty($_POST["gender"]))
    {
        $genderErr = "Gender is required";
    }
}
```

## PHP PROGRAMMING

---

```
    }
    else
    {
        $gender = test_input($_POST["gender"]);
    }
}
function test_input($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<h2>Absolute classes registration</h2>
    <p><span class = "error">* required field.</span></p>
    <form method = "post"
        action                =                "<?php                echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    <table>
        <tr>
            <td>Name:</td>
            <td><input type = "text" name = "name">
                <span        class        =        "error">*        <?php        echo
$nameErr;?></span>
            </td>
        </tr>
        <tr>
            <td>E-mail: </td>
            <td><input type = "text" name = "email">
                <span        class        =        "error">*        <?php        echo
$emailErr;?></span>
            </td>
        </tr>
        <tr>
            <td>Time:</td>
            <td> <input type = "text" name = "website">
                <span class = "error"><?php echo $websiteErr;?></span>
            </td>
        </tr>
        <tr>
            <td>Classes:</td>
```

## PHP PROGRAMMING

---

```

        <td> <textarea name = "comment" rows = "5" cols =
"40"></textarea></td>
    </tr>
    <tr>
        <td>Gender:</td>
        <td>
            <input type = "radio" name = "gender" value =
"female">Female
            <input type = "radio" name = "gender" value =
"male">Male
            <span class = "error">* <?php echo
$genderErr;?></span>
        </td>
    </tr>
    <td>
        <input type = "submit" name = "submit" value = "Submit">
    </td>
</table>
</form>
<?php
    echo "<h2>Your given values are as:</h2>";
    echo $name;
    echo "<br>";
    echo $email;
    echo "<br>";
    echo $website;
    echo "<br>";
    echo $comment;
    echo "<br>";
    echo $gender;
?>
</body>
</html>
```

**die():** It is an inbuilt function in PHP. It is used to print message and exit from the current php script. It is equivalent to exit() function in PHP.

```
<?php
$site = "";
fopen($site, "r")
or die("Unable to connect to given site.");
?>
```

**Output:** Unable to connect to given site.

# PHP PROGRAMMING

---

## CONSTRAINTS IN MYSQL

- ❖ MySQL CONSTRAINT is used to define rules to allow or restrict what values can be stored in columns. The purpose of inducing constraints is to enforce the integrity of a database.
- ❖ MySQL CONSTRAINTS are used to limit the type of data that can be inserted into a table.
- ❖ MySQL CONSTRAINTS can be classified into two types - column level and table level.
- ❖ The column level constraints can apply only to one column whereas table level constraints are applied to the entire table.
- ❖ MySQL CONSTRAINT is declared at the time of creating a table.
- ❖ Constrains are used to apply some conditions on tables. MySQL is supporting different types of constraints.

### 1. NOT NULL

MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table.

#### Example:

```
CREATE TABLE Persons ( ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255) NOT NULL, Age int );
```

### 2. UNIQUE

The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.

#### Example:

```
CREATE TABLE Persons ( ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int, UNIQUE (ID) );
```

### 3. PRIMARY KEY

---

## PHP PROGRAMMING

---

A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster.

**Example:**

```
CREATE TABLE Persons ( ID int NOT NULL, LastName varchar(255) NOT NULL,
FirstName varchar(255), Age int, PRIMARY KEY (ID) );
```

### 4. AUTO\_INCREMENT

AUTO\_INCREMENT allows a unique number to be generated automatically when a new record is inserted into a table. Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

**Example:**

```
CREATE TABLE Persons ( ID int NOT NULL AUTO_INCREMENT,
                        LastName varchar(255) NOT NULL,
                        FirstName varchar(255), Age int,
PRIMARY KEY (ID) );
```

### 5. DEFALUT

The DEFAULT constraint is used to provide a default value for a column. The default value will be added to all new records IF no other value is specified.

**Example:**

```
CREATE TABLE Persons ( ID int NOT NULL, LastName varchar(255) NOT NULL,
                        FirstName varchar(255), Age int,
                        City varchar(255) DEFAULT 'Sandnes' );
```

### 6. FOREIGN KEY

A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.

**Example:**

## PHP PROGRAMMING

---

```
CREATE TABLE Orders (    OrderID int NOT NULL,        OrderNumber int NOT
NULL,

                        PersonID int,        PRIMARY KEY (OrderID),
                        FOREIGN      KEY      (PersonID)      REFERENCES
Persons(PersonID) );
```

### 7. CHECK

A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression.

#### Example:

```
CREATE TABLE Persons (    ID int NOT NULL,        LastName varchar(255) NOT
NULL,FirstName varchar(255),    Age int,    CHECK (Age>=18) );
```

**mysqli\_select\_db():** To is used to select the particular database from mysql.

This function returns TRUE on success, or FALSE on failure

**Syntax:** mysqli\_select\_db(connection,dbname);

**mysqli\_query():** To execute an SQL query against the database. On success it returns TRUE. FALSE on failure

**Syntax:** mysqli\_query(connection,query);

**mysqli\_error():**The mysqli\_error() function returns the last error description for the most recent function call, if any. It returns a string that describes the error. An empty string if no error occurred.

**Syntax:** mysqli\_error(connection);

**mysqli\_connect\_error():** This function returns the error description from the last connection error, if any. Returns a string that describes the error. NULL if no error occurred

**Syntax:** mysqli\_connect\_error();

## PHP PROGRAMMING

---

**mysql\_fetch\_row():**The **mysql\_fetch\_row()** function returns a row from a recordset as a numeric array. This function gets a row from the **mysql\_query()** function and returns an array on success, or FALSE on failure or when there are no more rows.

**mysql\_fetch\_row(data)**

### **mysql\_fetch\_assoc():**

This function fetches the record from result set and it returns output as an associative array. Key is a column names, values are column values.

**mysql\_fetch\_array():**The **mysql\_fetch\_array()** function returns a row from a recordset as an associative array and/or a numeric array. This function gets a row from the **mysql\_query()** function and returns an array on success, or FALSE on failure or when there are no more rows. It is combination of previous two functions reads a record from result set and returns output as an associative array, a numeric array, or both

### **mysql\_fetch\_object():**

It reads a record from result set and returns output as an object. Object is a collection of properties. Property names are column names and values are column values. This object belongs to stdClass.

**mysql\_num\_fields():**The To get total number of fields from result set.

**mysql\_fetch\_field():****mysql\_fetch\_field()** function returns an object containing information of a field from a recordset. This function gets field data from the **mysql\_query()** function and returns an object on success, or FALSE on failure or when there are no more rows.

To fetch complete information of a field and returns output as an object.

**mysql\_fetch\_field(\$result)**

### **mysql\_field\_type():**

To get the type of field from the result set.



## PHP PROGRAMMING

---

**mysql\_field\_name():**The **mysql\_field\_name()** function returns the name of a field in a recordset.

Returns the field name on success, or FALSE on failure.

To get the name of field from result set.

`mysql_field_name(data,field_offset)`

**mysql\_field\_len():**

To get the length of the field from result set.

**mysql\_get\_client\_info():**

To get version information of mysql database.

**mysql\_data\_seek():**

To locate result set pointer on specified record.

**mysql\_field\_seek():**

To locate result set pointer on specified column.

**mysql\_close():**

To close the opened mysql connections.

**mysql\_list\_dbs():**

To get list of databases available.

**mysql\_list\_tables():**

To get the list of tables available in a database.

<https://www.studentstutorial.com/php/php-update-multiple-row.php>