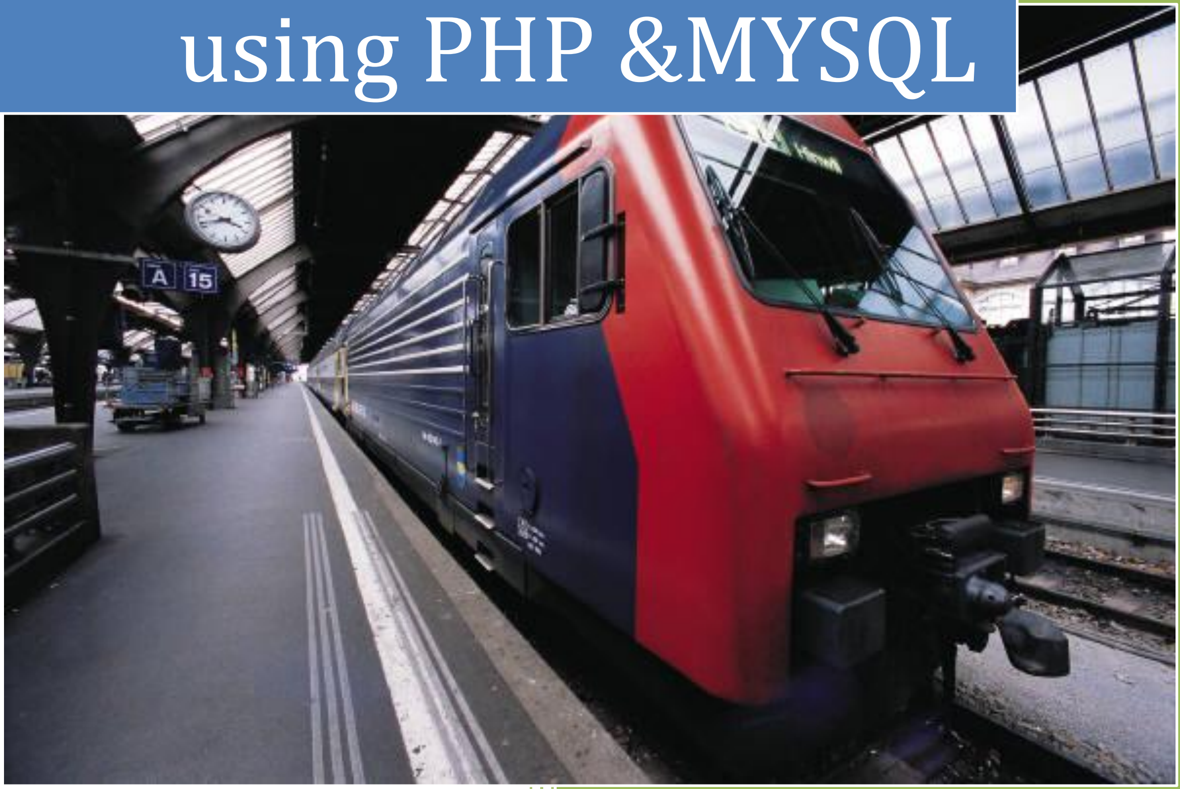
B.Sc

WebApplications Development using PHP &MYSQL



ABHYUDAYA MAHILA DEGREE COLLEGE

P.Y.KUMAR

B.Sc

WebApplications Development using PHP &MYSQL

Design, Implementation and Management

P. Y. Kumar ***M.C.A, M.Tech, M.Phil..,***
Krishna jyothi M.Sc,M.Tech,
a.kotaiah M.C.A,M.Tech
Head of the Department
Computer Science

This Book is dedicated to my Daughter. May God Bless you and be with you little one!

Web Applications Development using PHP & MYSQL

Unit-1 (10 hours)

The Building blocks of PHP: Variables, Data Types, Operators and Expressions, Constants. Flow Control Functions in PHP: Switching Flow, Loops, Code Blocks and Browser Output. Working with Functions: What is function?, Calling functions, Defining Functions, Returning the values from User-Defined Functions, Variable Scope, Saving state between Function calls with the static statement, more about arguments.

Unit-2: (10 hours)

Working with Arrays: What are Arrays? Creating Arrays, Some Array-Related Functions. Working with Objects: Creating Objects, Object Instance Working with Strings, Dates and Time: Formatting strings with PHP, Investigating Strings with PHP, Manipulating Strings with PHP, Using Date and Time Functions in PHP.

Unit-3: (10 hours)

Working with Forms: Creating Forms, Accessing Form Input with User defined Arrays, Combining HTML and PHP code on a single Page, Using Hidden Fields to save state, Redirecting the user, Sending Mail on Form Submission, and Working with File Uploads. Working with Cookies and User Sessions: Introducing Cookies, Setting a Cookie with PHP, Session Function Overview, Starting a Session, Working with session variables, passing session IDs in the Query String, Destroying Sessions and Unsetting Variables, Using Sessions in an Environment with Registered Users.

Unit-4: (10 hours)

Working with Files and Directories: Including Files with inclue(), Validating Files, Creating and Deleting Files, Opening a File for Writing, Reading or Appending, Reading from Files, Writing or Appending to a File, Working with Directories, Open Pipes to and from Process Using popen(), Running Commands with exec(), Running Commands with system() or passthru(). Working with Images: Understanding the Image-Creation Process, Necessary Modifications to PHP, Drawing a New Image, Getting Fancy with Pie Charts, Modifying Existing Images, Image Creation from User Input.

Unit-5: (10 hours)

Interacting with MySQL using PHP: MySQL Versus MySQLi Functions, Connecting to MySQL with PHP, Working with MySQL Data. Creating an Online Address Book: Planning and Creating Database Tables, Creating Menu, Creating Record Addition Mechanism, Viewing Records, Creating the Record Deletion Mechanism, Adding Sub-entities to a Record.

UNIT-1

Introduction to PHP:

PHP is a recursive acronym for "PHP: Hypertext Preprocessor". **Rasmus Lerdorf** unleashed the first version of PHP way back in 1994. 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.

PHP is a server side scripting language that is embedded in HTML. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server

PHP stands for Hypertext Preprocessor. PHP is a powerful and widely-used open source server-side scripting language to write dynamically generated web pages. PHP scripts are executed on the server and the result is sent to the browser as plain HTML.

PHP can be integrated with the number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

PHP can be embedded within normal HTML web pages. That means inside your HTML documents you'll have PHP statements like this:

Example

```
<html>
<head>
<title>PHP Application</title>
</head>
<body>
<?php
// Display greeting message
echo 'Hello World!';
?>
</body>
</html>
```

What You Can Do with PHP?

There are lot more things you can do with PHP.

- **You can generate dynamic pages and files.**
- **You can create, open, read, write and close files on the server.**
- **You can collect data from a web form such as user information, email, credit cardinformation and much more.**
- **You can send emails to the users of your website.**
- **You can send and receive cookies to track the visitor of your website.**
- **You can store, delete, and modify information in your database.**
- **You can restrict unauthorized access to your website.**
- **You can encrypt data for safe transmission over internet.**

What are the Advantages of PHP over Other Languages?

If you're familiar with other server-side languages like ASP.NET or JSP, you might be wondering what makes PHP so special. There are several advantages why one should choose PHP over other languages. Here are some of them:

- **Easy to learn:** PHP is easy to learn and use. For beginner programmers who just started out in web development, PHP is often considered as the best and preferable choice of scripting language to learn.
- **Open source:** PHP is an open-source project — the language is developed and maintained by a worldwide community of developers who make its source code freely available to download and use. There are no costs associated with using PHP for individual or commercial projects, including future updates.
- **Portability:** PHP runs on various platforms such as Microsoft Windows, Linux, Mac OS, etc. and it is compatible with almost all servers used today such Apache, IIS, etc.
- **Fast Performance:** Scripts written in PHP usually execute faster than those written in other scripting languages like ASP.NET or JSP.
- **Vast Community:** Since PHP is supported by the worldwide community, finding help or documentation for PHP online is extremely easy.

Explain about Standard PHP Syntax?

A PHP script starts with the <?php and ends with the ?> tag.

The PHP delimiter <?php and ?> in the following example simply tells the PHP engine to treat the enclosed code block as PHP code, rather than simple HTML.

Example

```
<?php
// Some code to be executed
echo "Hello, world!";
?>
```

Every PHP statement end with a semicolon (;) — this tells the PHP engine that the end of the current statement has been reached.

How to Embedding PHP within HTML?

PHP files are plain text files with .php extension. Inside a PHP file you can write HTML like you do in regular HTML pages as well as embed PHP codes for server side execution.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>A Simple PHP File</title>
</head>
<body>
<h1><?php echo "Hello, world!"; ?></h1>
</body>
</html>
```

The above example shows how you can embed PHP codes within HTML to create well-formed dynamic web pages. If you view the source code of the resulting web page in your browser, the only difference you will see is the PHP code <?php echo "Hello, world!"; ?> has been replaced with the output "Hello, world!".

What happened here is? when you run this code the PHP engine executed the instructions between the <?php … ?> tags and leave rest of the thing as it is. At the end the web server send the final output back to your browser which is completely in HTML.

Q) Explain about PHP Comments?

A comment is simply text that is ignored by the PHP engine. The purpose of comments is to make the code more readable. It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the PHP.

PHP support single-line as well as multi-line comments. To write a single-line comment either start the line with either two slashes (//) or a hash symbol (#). For example:

Example

```
<?php
// This is a single line comment
# This is also a single line comment
echo "Hello, world!";
?>
```

However to write multi-line comments, start the comment with a slash followed by an asterisk (/*) and end the comment with an asterisk followed by a slash (*/), like this:

Example

```
<?php
/*
This is a multiple line comment block
that spans across more than
one line
*/
echo "Hello, world!";
?>
```

Q) What is Variable in PHP?

Variables are used to store data, like string of text, numbers, etc. Variable values can change over the course of a script. Here're some important things to know about variables:

- In PHP, a variable does not need to be declared before adding a value to it. PHP automatically converts the variable to the correct data type, depending on its value.
- After declaring a variable it can be reused throughout the code.
- The assignment operator (=) used to assign value to a variable.

In PHP variable can be declared as: $var_name = value;

Example

```
<?php
// Declaring variables
$txt = "Hello World!";
$number = 10;
// Displaying variables value
echo $txt; // Output: Hello World!
echo $number; // Output: 10
?>
```

In the above example we have created two variables where first one has assigned with a string value and the second has assigned with a number. Later we've displayed the variables values in the browser using the echo statement. The PHP echo statement is often used to output data to the browser. We will learn more about this in upcoming chapter.

Naming Conventions for PHP Variables

These are the following rules for naming a PHP variable:

- **All variables in PHP start with a $ sign, followed by the name of the variable.**
- **A variable name must start with a letter or the underscore character _.**
- **A variable name cannot start with a number.**
- **A variable name in PHP can only contain alpha-numeric characters and underscores(A-z, 0-9, and _).**
- **A variable name cannot contain spaces.**

Q) What is Constant in PHP?

A constant is a name or an identifier for a fixed value. Constant are like variables except that one they are defined, they cannot be undefined or changed (except magic constants).

Constants are very useful for storing data that doesn't change while the script is running. Common examples of such data include configuration settings such as database username and password, website's base URL, company name, etc.

Constants are defined using PHP's define() function, which accepts two arguments: the name of the constant, and its value. Once defined the constant value can be accessed at any time just by referring to its name. Here is a simple example:

Example

```
<?php
// Defining constant
define("SITE_URL",  "https://www.tutorialrepublic.com/");
// Using constant
echo 'Thank you for visiting - ' . SITE_URL;
?>
```

The output of the above code will be:

Thank you for visiting - https://www.tutorialrepublic.com/

Q) Explain about Data Types in PHP?

The values assigned to a PHP variable may be of different data types including simple string and numeric types to more complex data types like arrays and objects.

PHP supports total eight primitive data types: Integer, Floating point number or Float, String, Booleans, Array, Object, resource and NULL. These data types are used to construct variables. Now let's discuss each one of them in detail.

PHP Integers

Integers are whole numbers, without a decimal point (..., -2, -1, 0, 1, 2, ...). Integers can be specified in decimal (base 10), hexadecimal (base 16 - prefixed with 0x) or octal (base 8 - prefixed with 0) notation, optionally preceded by a sign (- or +).

Example

```
<?php
$a = 123; // decimal number
var_dump($a);
echo "<br>";
$b = -123; // a negative number
var_dump($b);
echo "<br>";
$c = 0x1A; // hexadecimal number
var_dump($c);
echo "<br>";
$d = 0123; // octal number
var_dump($d);
?>
```

PHP Strings

Strings are sequences of characters, where every character is the same as a byte.

A string can hold letters, numbers, and special characters and it can be as large as up to 2GB (2147483647 bytes maximum). The simplest way to specify a string is to enclose it in single quotes (e.g. 'Hello world!'), however you can also use double quotes ("Hello world!").

Example

```
<?php
$a = 'Hello world!';
echo $a;
echo "<br>";
$b = "Hello world!";
echo $b;
echo "<br>";
$c = 'Stay here, I\'ll be back.';
echo $c;
?>
```

PHP Floating Point Numbers or Doubles

Floating point numbers (also known as "floats", "doubles", or "real numbers") are decimal or fractional numbers, like demonstrated in the example below.

Example

```
<?php
$a = 1.234;
var_dump($a);
echo "<br>";
$b = 10.2e3;
var_dump($b);
echo "<br>";
$c = 4E-10;
var_dump($c);
?>
```

PHP Booleans

Booleans are like a switch it has only two possible values either 1 (true) or 0 (false).

Example

```
<?php
// Assign the value TRUE to a variable
$show_error = true;
var_dump($show_error);
?>
```

PHP Arrays

An array is a variable that can hold more than one value at a time. It is useful to aggregate a series of related items together, for example a set of country or city names.

An array is formally defined as an indexed collection of data values. Each index (also known as the key) of an array is unique and references a corresponding value.

Example

```
<?php
$colors = array("Red", "Green", "Blue");
var_dump($colors);
echo "<br>";
$color_codes = array(
"Red" => "#ff0000",
"Green" => "#00ff00",
"Blue" => "#0000ff"
);
var_dump($color_codes);
?>
```

PHP Objects

An object is a data type that not only allows storing data but also information on, how to process that data. An object is a specific instance of a class which serve as templates for objects. Objects are created based on this template via the new keyword.

Every object has properties and methods corresponding to those of its parent class. Every object instance is completely independent, with its own properties and methods, and can thus be manipulated independently of other objects of the same class.

Here's a simple example of a class definition followed by the object creation.

Example

```
<?php
// Class definition
class greeting{
// properties
public $str = "Hello World!";
// methods
function show_greeting(){
return $this->str;
}
}
// Create object from class
$message = new greeting;
var_dump($message);
?>
```

PHP NULL

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data. NULL is the only possible value of type null.

Example

```
<?php
$a = NULL;
var_dump($a);
echo "<br>";
$b = "Hello World!";
$b = NULL;
var_dump($b);
?>
```

When a variable is created without a value in PHP like $var; it is automatically assigned a value of null. Many novice PHP developers mistakenly considered both $var1 = NULL; and $var2 = ""; are same, but this is not true. Both variables are different — the $var1 has null value while $var2 indicates no value assigned to it.

PHP Resources
A resource is a special variable, holding a reference to an external resource.
Resource variables typically hold special handlers to opened files and database connections.

Example

```
<?php
// Open a file for reading
$handle = fopen("note.txt", "r");
var_dump($handle);
echo "<br>";
// Connect to MySQL database server with default setting
$link = mysql_connect("localhost", "root", "");
var_dump($link);
?>
```

Q) What is Operators in PHP?

Operators are symbols that tell the PHP processor to perform certain actions. For example, the addition (+) symbol is an operator that tells PHP to add two variables or values, while the greater-than (>) symbol is an operator that tells PHP to compare two values.

The following lists describe the different operators used in PHP.

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.
*	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

The following example will show you these arithmetic operators in action:

Example

```
<?php
$x = 10;
$y = 4;
echo($x + $y); // 0utputs: 14
echo($x - $y); // 0utputs: 6
echo($x * $y); // 0utputs: 40
echo($x / $y); // 0utputs: 2.5
echo($x % $y); // 0utputs: 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

The following example will show you these assignment operators in action:

Example

```
<?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	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
<	Less than	$x < $y	True if $x is less than $y
>	Greater than	$x > $y	True if $x is greater than $y
>=	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

The following example will show you these comparison operators in action:

Example

```
<?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-decrement	Returns $x, then decrements $x by one

The following example will show you these increment and decrement operators in action:

Example

```
<?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	rue if both $x and $y are true

||	Or	$x || $y	True if either $$x or $y is true
!	Not	!$x	True if $x is not true

The following example will show you these logical operators in action:

Example

```
<?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**
.	Concatenation	$str1 . $str2	Concatenation of $str1 and $str2
.=	Concatenation assignment	$str1 .= $str2	Appends the $str2 to the $str1

The following example will show you these string operators in action:

Example

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

PHP Array Operators

The array operators are used to compare arrays:

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

The following example will show you these array operators in action:

Example

```
<?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
?>
```

PHP Spaceship Operator PHP 7

PHP 7 introduces a new spaceship operator (<=>) which can be used for comparing two expressions. It is also known as combined comparison operator.

The spaceship operator returns 0 if both operands are equal, 1 if the left is greater, and -1 if the right is greater. It basically provides three-way comparison as shown in the following table:

Operator	**<=> Equivalent**
$x < $y	($x <=> $y) === -1
$x <= $y	($x <=> $y) === -1 \|\| ($x <=> $y) === 0

Operator	**<=> Equivalent**
$x == $y	($x <=> $y) === 0
$x != $y	($x <=> $y) !== 0
$x >= $y	($x <=> $y) === 1 \|\| ($x <=> $y) === 0
$x > $y	($x <=> $y) === 1

The following example will show you how spaceship operator actually works:

Example

```
<?php
// Comparing Integers
echo 1 <=> 1; // Outputs: 0
echo 1 <=> 2; // Outputs: -1
echo 2 <=> 1; // Outputs: 1
// Comparing Floats
echo 1.5 <=> 1.5; // Outputs: 0
echo 1.5 <=> 2.5; // Outputs: -1
echo 2.5 <=> 1.5; // Outputs: 1
// Comparing Strings
echo "x" <=> "x"; // Outputs: 0
echo "x" <=> "y"; // Outputs: -1
echo "y" <=> "x"; // Outputs: 1
?>
```

Q) Explain about PHP Conditional Statements?

Like most programming languages, PHP also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time. This means, you can create test conditions in the form of expressions that evaluates to either true or false and based on these results you can perform certain actions.

There are several statements in PHP that you can use to make decisions:

- The **if** statement
- The **if...else** statement
- The **if...elseif....else** statement
- The **switch.. case** statement

We will explore each of these statements in the coming sections.

The 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:

```
if(condition)
{
   // Code to be executed
}
```

The following example will output "Have a nice weekend!" if the current day is Friday:

Example

```
<?php
$d = date("D");
if($d == "Fri"){
echo "Have a nice weekend!";
}
?>
```

The 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:

```
if(condition){
   // Code to be executed if condition is true
} else{
   // Code to be executed if condition is false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!"

Example

```
<?php
$d = date("D");
if($d == "Fri"){
echo "Have a nice weekend!";
} else{
echo "Have a nice day!";
}
?>
```

The if...elseif...else Statement

The *if...elseif...else* a special statement that is used to combine multiple *if...else* statements.

```
if(condition){
   // Code to be executed if condition is true
} elseif(condition){
   // Code to be executed if condition is true
} else{
   // Code to be executed if condition is false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday, otherwise it will output "Have a nice day!"

Example

```
<?php
$d = date("D");
if($d == "Fri"){
echo "Have a nice weekend!";
} elseif($d == "Sun"){
echo "Have a nice Sunday!";
} else{
echo "Have a nice day!";
}
?>
```

The Ternary Operator

The ternary operator provides a shorthand way of writing the *if...else* statements. The ternary operator is represented by the question mark (?) symbol and it takes three operands: a condition to check, a result for ture, and a result for false.

To understand how this operator works, consider the following examples:

Example

```
<?php
if($age < 18){
echo 'Child'; // Display Child if age is less than 18
} else{
echo 'Adult'; // Display Adult if age is greater than or equal to 18
}
?>
```

Using the ternary operator the same code could be written in a more compact way:

Example

```
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

The ternary operator in the example above selects the value on the left of the colon (i.e. 'Child') if the condition evaluates to true (i.e. if $age is less than 18), and selects the value on the right of the colon (i.e. 'Adult') if the condition evaluates to false.

Switch…Case

The switch-case statement is an alternative to the if-elseif-else statement, which does almost the same thing. The switch-case statement tests a variable against a series of values until it finds a match, and then executes the block of code corresponding to that match.

```
switch(n){
  case
  label1:
    // Code to be executed if
    n=label1break;
  case label2:
    // Code to be executed if
    n=label2break;
  ...
  default:
    // Code to be executed if n is different from all labels
}
```

Consider the following example, which display a different message for each day.

Example

```
<?php
$today = date("D");
switch($today){
case "Mon":
echo "Today is Monday. Clean your house.";
break;
case "Tue":
echo "Today is Tuesday. Buy some food.";
break;
case "Wed":
echo "Today is Wednesday. Visit a doctor.";
break;
case "Thu":
echo "Today is Thursday. Repair your car.";
break;
case "Fri":
echo "Today is Friday. Party tonight.";
break;
case "Sat":
echo "Today is Saturday. Its movie time.";
break;
case "Sun":
echo "Today is Sunday. Do some rest.";
break;
default:
echo "No information available for that day.";
break;
}
?>
```

The switch-case statement differs from the if-elseif-else statement in one important way. The switch statement executes line by line (i.e. statement by statement) and once PHP finds a case statement that evaluates to true, it's not only executes the code corresponding to that case statement, but also executes all the subsequent case statements till the end of the switch block automatically.

To prevent this add a break statement to the end of each case block. The break statement tells PHP to break out of the switch-case statement block once it executes the code associated with the first true case.

Q) Explain Different Types of Loops in PHP ?

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.

- **while** — loops through a block of code until the condition is evaluate to true.
- **do…while** — the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.
- **for** — loops through a block of code until the counter reaches a specified number.
- **foreach** — loops through a block of code for each element in an array.

You will also learn how to loop through the values of array using foreach() loop at the end of this chapter. The foreach() loop work specifically with arrays.

PHP while Loop

The while statement will loops through a block of code until the condition in the while statement evaluate to true.

```
while(condition){
   // Code to be executed
}
```

The example below define a loop that starts with $i=1. The loop will continue to run as long as $i is less than or equal to 3. The $i will increase by 1 each time the loop runs:

Example

```
<?php
$i = 1;
while($i <= 3){
$i++;
echo "The number is " . $i . "<br>";
}
?>
```

PHP do…while Loop

The do-while loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a do-while loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

```
do{
   // Code to be executed
}
while(condition);
```

The following example define a loop that starts with $i=1. It will then increase $i with 1, and print the output. Then the condition is evaluated, and the loop will continue to run as long as $i is less than, or equal to 3.

Example

```
<?php
$i = 1;
do{
$i++;
echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

Difference Between while and do…while Loop

The while loop differs from the do-while loop in one important way — with a while loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a do-while loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

PHP for Loop

The for loop repeats a block of code until a certain condition is met. It is typically used to execute a block of code for certain number of times.

```
for(initialization; condition; increment){
   // Code to be executed
}
```

The parameters of for loop have following meanings:

- ☞ initialization — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.
- ☞ condition — in the beginning of each iteration, condition is evaluated. If it evaluates to true, the loop continues and the nested statements are executed. If it evaluates to false, the execution of the loop ends.
- ☞ increment — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

The example below defines a loop that starts with $i=1. The loop will continued until $i is less than, or equal to 5. The variable $i will increase by 1 each time the loop runs:

Example

```
<?php
for($i=1; $i<=3; $i++){
echo "The number is " . $i . "<br>";
}
?>
```

PHP foreach Loop

The foreach loop is used to iterate over arrays.

```
foreach($array as $value){
   // Code to be executed
}
```

The following example demonstrates a loop that will print the values of the given array:

Example

```
<?php
$colors = array("Red", "Green", "Blue");
// Loop through colors array
foreach($colors as $value){
echo $value . "<br>";
}
?>
```

There is one more syntax of foreach loop, which is extension of the first.

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

Example

```
<?php
$superhero = array(
"name" => "Peter Parker",
"email" => "peterparker@mail.com",
"age" => 18
);
// Loop through superhero array
foreach($superhero as $key => $value){
echo $key . " : " . $value . "<br>";
}
?>
```

Q) Explain about functions in

PHP?PHP Built-in Functions

A function is a self-contained block of code that performs a specific task.

PHP has a huge collection of internal or built-in functions that you can call directly within your PHP scripts to perform a specific task, like gettype(), print_r(), var_dump, etc.

Please check out PHP reference section for a complete list of useful PHP built-in functions.

PHP User-Defined Functions

In addition to the built-in functions, PHP also allows you to define your own functions. It is a way to create reusable code packages that perform specific tasks and can be kept and maintained separately form main program. Here are some advantages of using functions:

- ☞**Functions reduces the repetition of code within a program** — Function allows you to extract commonly used block of code into a single component. Now can you can perform the same task by calling this function wherever you want without having to copy and paste the same block of code again and again.
- ☞**Functions makes the code much easier to maintain** — Since a function created once can be used many times, so any changes made inside a function automatically implemented at all the places without touching the several files.
- ☞**Functions makes it easier to eliminate the errors** — When the program is subdivided into functions, if any error occur you know exactly what function causing the error and where to find it. Therefore, fixing errors becomes much easier.
- ☞**Functions can be reused in other application** — Because a function is separated from the rest of the script, it's easy to reuse the same function in other applications just by including the php file containing those functions.

The following section will show you how easily you can define your own function in PHP.

Creating and Invoking Functions

The basic syntax of creating a custom function can be give with:

```
function functionName(){
   // Code to be executed
}
```

The declaration of a user-defined function start with the word function, followed by the name of the function you want to create followed by parentheses i.e. () and finally place your function's code between curly brackets {}.

This is a simple example of an user-defined function, that disply today's date:

Example

```
<?php
// Defining function
function whatIsToday(){
echo "Today is " . date('l', mktime());
}
// Calling function
whatIsToday();
?>
```

Note:A function name must start with a letter or underscore character not with a number, optionally followed by the more letters, numbers, or underscore characters. Function names are case-insensitive.

Functions with Parameters

You can specify parameters when you define your function to accept input values at run time. The parameters work like placeholder variables within a function; they're replaced at run time by the values (known as argument) provided to the function at the time of invocation.

```
function myFunc($oneParameter, $anotherParameter){
   // Code to be executed
}
```

You can define as many parameters as you like. However for each parameter you specify, a corresponding argument needs to be passed to the function when it is called.

The getSum() function in following example takes two integer values as arguments, simply add them together and then display the result in the browser.

Example

```
<?php
// Defining function
function getSum($num1, $num2){
$sum = $num1 + $num2;
echo "Sum of the two numbers $num1 and $num2 is : $sum";
}
// Calling function
getSum(10, 20);
?>
```

The output of the above code will be:

```
Sum of the two numbers 10 and 20 is : 30
```

Functions with Optional Parameters and Default Values

You can also create functions with optional parameters — just insert the parameter name, followed by an equals (=) sign, followed by a default value, like this.

Example

```
<?php
// Defining function
function customFont($font, $size=1.5){
 echo "<p style=\"font-family: $font; font-size: {$size}em;\">Hello,
 world!</p>";
}
// Calling function
customFont("Arial", 2);
customFont("Times", 3);
customFont("Courier");
?>
```

As you can see the third call to customFont() doesn't include the second argument. This causes PHP engine to use the default value for the $size parameter which is 1.5.

Returning Values from a Function

A function can return a value back to the script that called the function using the return statement. The value may be of any type, including arrays and objects.

Example

```
<?php
// Defining function
function getSum($num1, $num2){
 $total = $num1 + $num2;
 return $total;
}
// Printing returned value
echo getSum(5, 10); // Outputs: 15
?>
```

A function can not return multiple values. However, you can obtain similar results by returning an array, as demonstrated in the following example.

Passing Arguments to a Function by Reference

In PHP there are two ways you can pass arguments to a function: *by value* and *by reference*. By default, function arguments are passed by value so that if the value of the argument within the function is changed, it does not get affected outside of the function. However, to allow a function to modify its arguments, they must be passed by reference.

Passing an argument by reference is done by prepending an ampersand (&) to the argument name in the function definition, as shown in the example below:

Example

```
<?php
/* Defining a function that multiply a number
by itself and return the new value */
function selfMultiply(&$number){
 $number *= $number;
 return $number;
}
$mynum = 5;
echo $mynum; // Outputs: 5
selfMultiply($mynum);
echo $mynum; // Outputs: 25
?>
```

Q) Understanding the Variable Scope

However, you can declare the variables anywhere in a PHP script. But, the location of the declaration determines the extent of a variable's visibility within the PHP program i.e. where the variable can be used or accessed. This accessibility is known as *variable scope*.

By default, variables declared within a function are local and they cannot be viewed or manipulated from outside of that function, as demonstrated in the example below:

Example

```
<?php
// Defining function
function test(){
 $greet = "Hello World!";
 echo $greet;
}
test(); // Outputs: Hello World!
echo $greet; // Generate undefined variable error
?>
```

Similarly, if you try to access or import an outside variable inside the function, you'll get an undefined variable error, as shown in the following example:

Example

```
<?php
$greet = "Hello World!";
// Defining function
function test(){
 echo $greet;
}
test(); // Generate undefined variable error
echo $greet; // Outputs: Hello World!
?>
```

As you can see in the above examples the variable declared inside the function is not accessible from outside, likewise the variable declared outside of the function is not accessible inside of the function. This separation reduces the chances of variables within a function getting affected by the variables in the main program.

The global Keyword

There may be a situation when you need to import a variable from the main program into a function, or vice versa. In such cases, you can use the global keyword before the variables inside a function. This keyword turns the variable into a global variable, making it visible or accessible both inside and outside the function, as show in the example below:

Example

```
<?php
$greet = "Hello World!";
 // Defining function
function test(){
global $greet;
echo $greet;
}
test(); // Outpus: Hello World!
echo $greet; // Outpus: Hello World!
// Assign a new value to variable
$greet = "Goodbye";
test(); // Outputs: Goodbye
echo $greet; // Outputs: Goodbye
?>
```

Q) Explain about PHP arrays?

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

- ☞ **Numeric array** − An array with a numeric index. Values are stored and accessed in linear fashion.
- ☞ **Associative array** − An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- ☞ **Multidimensional array** − An array containing one or more arrays and values are accessed using multiple indices

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
  <body>
    <?php
      /* First method to create array. */
      $numbers = array( 1, 2, 3, 4, 5);

      foreach( $numbers as
        $value ) {echo "Value is
        $value <br />";
      }

      /* Second method to create array. */
      $numbers[0] = "one";
      $numbers[1] = "two";
      $numbers[2] = "three";
      $numbers[3] = "four";
      $numbers[4] = "five";

      foreach( $numbers as
        $value ) {echo "Value is
        $value <br />";
      }
   ?>

  </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is
threeValue
is four
Value is five
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE − Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
  <body>

    <?php
      /* First method to associate create array. */
      $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);

      echo "Salary of mohammad is ".
      $salaries['mohammad'] . "<br />";echo "Salary
      of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ". $salaries['zara']. "<br />";

      /* Second method to create array. */
      $salaries['mohammad'] = "high";
      $salaries['qadir'] = "medium";
      $salaries['zara'] = "low";

      echo "Salary of mohammad is ".
      $salaries['mohammad'] . "<br />";echo "Salary
      of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ". $salaries['zara']. "<br />";
    ?>

  </body>
</html>
```

This will produce the following result −

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

Multidimensional Arrays

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

Example

In this example we create a two dimensional array to store marks of three students in three subjects −

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
  <body>

    <?php
      $marks = array(
        "mohammad" =>
        array (
          "physics" => 35,
          "maths" => 30,
          "chemistry" => 39
        ),

        "qadir" =>
          array (
          "physics"
          => 30,
          "maths" => 32,
          "chemistry" => 29
        ),

        "zara" =>
          array (
          "physics"
          => 31,
          "maths" => 22,
          "chemistry" => 39
        )
     );

      /* Accessing multi-dimensional
      array values */echo "Marks for
      mohammad in physics : " ;
      echo $marks['mohammad']['physics'] . "<br />";

      echo "Marks for qadir in maths : ";
      echo $marks['qadir']['maths'] . "<br />";

      echo "Marks for zara in chemistry : " ;
      echo $marks['zara']['chemistry'] . "<br />";
    ?>
  </body>
</html>
```

This will produce the following result −

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

Q) Some Array-Related Functions ?

More than 70 array-related functions are built into PHP, which you can read about in detail at http://www.php.net/array. Some of the more common (and useful) functions are explained in this section.

- ✓ count() and sizeof() Each of these functions counts the number of elements in an array. Given the following array

```
$colors = array("blue", "black", "red", "green");
```

both `count($colors);` and `sizeof($colors);` return a value of 4.

- ✓ `each() and list()` These functions usually appear together, in the context of stepping through an array and returning its keys and values. You saw an example of this previously, where we stepped through the `$c` array and printed its contents.
- ✓ `foreach()` This function is also used to step through an array, assigning the value of an element to a given variable, as you saw in the previous section.
- ✓ `reset()` This function rewinds the pointer to the beginning of an array, as in this example:

```
reset($character);
```

This function is useful when you are performing multiple manipulations on an array, such as sorting, extracting values, and so forth.

- ✓ `array_push()` This function adds one or more elements to the end of an existing array, as in this example:

```
array_push($existingArray, "element 1", "element 2", "element
3");
```

- ✓ `array_pop()` This function removes (and returns) the last element of an existing array, as in this example:

```
$last_element = array_pop($existingArray);
```

- ✓ `array_unshift()` This function adds one or more elements to the beginning of an existing array, as in this example:

```
array_unshift($existingArray, "element 1", "element 2",
"element 3");
```

- ✓ `array_shift()` This function removes (and returns) the first element of an existing array, as in this example, where the value of the element in the first position of `$existingArray` is assigned to the variable `$first_element`:

```
$first_element = array_shift($existingArray);
```

- ✓ `array_merge()` This function combines two or more existing arrays, as in this example:

```
$newArray = array_merge($array1, $array2);
```

- ✓ `array_keys()` This function returns an array containing all the key names within a given array, as in this example:

```
$keysArray = array_keys($existingArray);
```

- ✓ `array_values()` This function returns an array containing all the values within a given array, as in this example:

```
$valuesArray = array_values($existingArray);
```

- ✓ `shuffle()` This function randomizes the elements of a given array. The syntax of this function is simply as follows:

```
shuffle($existingArray);
```

Q) Explain about PHP Strings?

A string is a collection of characters which is enclosed with in a double quotation or single quotation. String contains digits, alphabets and special symbols.

PHP String Functions

we will look at some commonly used functions to manipulate strings.

Get The Length of a String

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

The example below returns the length of the string "Hello world!":

Example

```
<html>
<body>
<?php
echo strlen("Hello world!");
?>
</body>
</html>
```

The output of the code above will be: 12.

Count The Number of Words in a String

The PHP str_word_count() function counts the number of words in a string:

Example

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

The output of the code above will be: 2.

Reverse a String

The PHP strrev() function reverses a string:

Example

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

The output of the code above will be: !dlrow olleH.

Replace Text Within a String

The PHP str_replace() function replaces some characters with some other characters in a string.
The example below replaces the text "world" with "Dolly":

```
<html>
<body>
<?php
echo str_replace("world", "Dolly", "Hello world!");
?>
</body>
</html>
```

The output of the code above will be: Hello Dolly!

PHP str_replace() Function

Example

Replace the characters "world" in the string "Hello world!" with "Peter":

```
<html>
<body>
<?php
 echo str_replace("world","Peter","Hello world!");
?>
</body>
</html>
```

<p>In this example, we search for the string "Hello World!", find the value "world" and then replace the value with "Peter".</p>

The output of the code above will be: Hello Peter!

In this example, we search for the string "Hello World!", find the value "world" and then replace the value with "Peter".

Definition and Usage

The str_replace() function replaces some characters with some other characters in a string.

This function works by the following rules:

- **If the string to be searched is an array, it returns an array**
- **If the string to be searched is an array, find and replace is performed with every array element**
- **If both find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace**
- **If find is an array and replace is a string, the replace string will be used for every find value**

Note: This function is case-sensitive. Use the str_ireplace() function to perform a case-insensitive search.

Note: This function is binary-safe.

Syntax

str_replace(*find,replace,string,count*)

Parameter	Description
find	Required. Specifies the value to find
replace	Required. Specifies the value to replace the
value in *findstring*	Required. Specifies the string to be searched
count	Optional. A variable that counts the number of replacements

PHP str_repeat() Function

The str_repeat() function repeats a string a specified number of times.

Syntax

str_repeat(*string,repeat*)

Parameter	Description
string	*Required. Specifies the string to repeat*
repeat	*Required. Specifies the number of times the string will be repeated. Must be greater or equal to 0*

Example:

```
<html>
<body>
<?php echo str_repeat("shashish",8);
?>
</body>
</html>
```

PHP str_shuffle() Function

Definition and Usage

The str_shuffle() function randomly shuffles all the characters of a string.

Syntax

str_shuffle(*string*)

Parameter	Description
string	Required. Specifies the string to shuffle

```
<html>
<body>
<?php
echo str_shuffle("Hello World");
?>
<p>Try to refresh the page. This function will randomly shuffle all characters
each time.</p>
</body>
</html>
```

Output:

```
dolerW Hllo
```

Try to refresh the page. This function will randomly shuffle all characters each time.

PHP str_split() Function

Definition and Usage

The str_split() function splits a string into an array.

Syntax

str_split(*string,length*)

Parameter	Description
string	Required. Specifies the string to split
length	Optional. Specifies the length of each array element. Default is 1

```
<html>
<body>
<?php print_r(str_split("Hello"));
?>
</body>
</html>
```

Output:

```
Array ( [0] => H [1] => e [2] => l [3] => l [4] => o )
```

PHP str_word_count() Function

Definition and Usage

The str_word_count() function counts the number of words in a string.

Syntax

str_word_count(string,return,char)

Parameter	Description
string	Required. Specifies the string to check
return	Optional. Specifies the return value of the str_word_count() function. **Possible values:** • 0 - Default. Returns the number of words found • 1 - Returns an array with the words from the string • 2 - Returns an array where the key is the position of the word in thestring, and value is the actual word
char	Optional. Specifies special characters to be considered as words.

Example:

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

Output:

```
2
```

PHP strcasecmp() Function

Definition and Usage

The strcasecmp() function compares two strings.

Syntax

strcasecmp(*string1*,*string2*)

Parameter	Description
string1	Required. Specifies the first string to compare
string2	Required. Specifies the second string to compare

Example:

```
<html>
<body>
<?php
echo strcasecmp("Hello world!","HELLO WORLD!");
?>
<p>If this function returns 0, the two strings are equal.</p>
</body>
</html>
```

Output:

```
0

If this function returns 0, the two strings are equal.
```

PHP strcmp() Function

Definition and Usage

The strcmp() function compares two strings.

Syntax

strcmp(*string1*,*string2*)

Parameter	Description
string1	Required. Specifies the first string to compare
string2	Required. Specifies the second string to compare

```
<html>
<body>

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

<p>If this function returns 0, the two strings are equal.</p>

</body>
</html>
```

Output:

```
0

If this function returns 0, the two strings are equal.
```

PHP strtolower() Function

Definition and Usage

The strtolower() function converts a string to lowercase.

Syntax

strtolower(*string*)

Parameter	Description
string	Required. Specifies the string to convert

```
<html>
<body>
<?php
echo strtolower("Hello WORLD.");
?>
</body>
</html>
```

Output:

```
hello world.
```

PHP strtoupper() Function

Definition and Usage

The strtoupper() function converts a string to uppercase.

Syntax

strtoupper(*string*)

Parameter	**Description**
string	**Required. Specifies the string to convert**

```
<html>
<body>
<?php
echo strtoupper("Hello WORLD!");
?>
</body>
</html>
```

Output:

```
HELLO WORLD!
```

PHP substr() Function

Definition and Usage

The substr() function returns a part of a string.

Syntax

substr(*string,start,length*)

Parameter	**Description**
string	Required. Specifies the string to return a part of
start	Required. Specifies where to start in the string ● A positive number - Start at a specified position in the string ● A negative number - Start at a specified position from the end of thestring ● 0 - Start at the first character in string
length	Optional. Specifies the length of the returned string. Default is to the end of the string. ● A positive number - The length to be returned from the start parameter ● Negative number - The length to be returned from the end of the string

```
<html>
<body>
<?php
echo substr("Hello world",6);
?>
</body>
</html>
```

Output:

```
World
```

PHP provides many functions that will transform a string argument, subtly or radically, as you'll soon see.

Cleaning Up a String with `trim()`, `ltrim()`, and `strip_tags()`

When you acquire text from user input or an external file, you can't always be sure that you haven't also picked up white space at the beginning and end of your data.

The `trim()` function shaves any white space characters, including newlines, tabs, and spaces, from both the start and end of a string. It accepts the string to be modified, returning the cleaned-up version. For example:

```
<?php
 $text = "\t\tlots of room to breathe      ";
echo "<pre>$text</pre>";
// prints "        lots of room to breathe      ";
 $text = trim($text);
echo "<pre>$text</pre>";
 // prints "lots of room to breathe";
 ?>
```

Q) Formatting Strings with PHP?

PHP provides two functions that allow you first to apply formatting, whether to round doubles to a given number of decimal places, define alignment within a field, or display data according to different number systems. In this section, you will look at a few of the formatting options provided by `printf()` and `sprintf()`.

Working with `printf()`

If you have any experience with a C-like programming language, you will be familiar with the concept of the `printf()` function. The `printf()` function requires a string argument, known as a format control string. It also accepts additional arguments of different types, which you'll learn about in a moment. The format control string contains instructions regarding the display of these additional arguments. The following fragment, for example, uses `printf()` to output an integer as an octal (or base-8) number:

```
<?php printf("This is my number: %o", 55); // prints "This is my number:
67" ?>
```

Within the format control string (the first argument), we have included a special code, known as a conversion specification.

A conversion specification begins with a percent (`%`) symbol and defines how to treat the corresponding argument to `printf()`. You can include as many conversion specifications as you want within the format control string, as long as you send an equivalent number of arguments to `printf()`.

The following fragment outputs two floating-point numbers using `printf()`:

```
<?php printf("First number: %f<br/>Second number: %f<br/>", 55, 66); //
Prints: // First number: 55.000000 // Second number: 66.000000 ?>
```

The first conversion specification corresponds to the first of the additional arguments to `printf()`, or 55. The second conversion specification corresponds to 66. The `f` following the percent symbol requires that the data be treated as a floating-point number. This part of the conversion specification is the type specifier.

`printf()` and Type Specifiers

You have already come across two type specifiers, `o`, which displays integers as octals, and f, which displays integers as floating-point numbers. Table 10.1 lists the other type specifiers available.

Table 10.1. Type Specifiers

Specifier	Description
`d`	Display argument as a decimal number
`b`	Display an integer as a binary number
`c`	Display an integer as ASCII equivalent
`f`	Display an integer as a floating-point number (double)
`o`	Display an integer as an octal number (base 8)
`s`	Display argument as a string
`x`	Display an integer as a lowercase hexadecimal number (base 16)
`X`	Display an integer as an uppercase hexadecimal number (base 16)

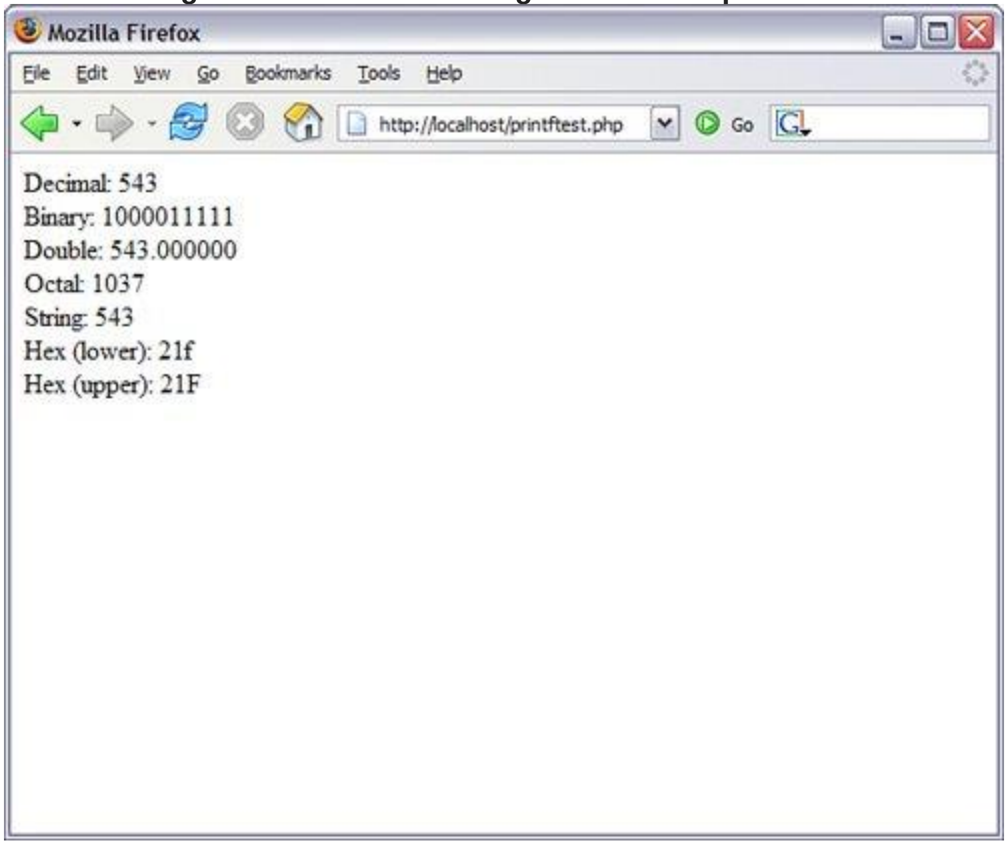
Listing 10.1 uses `printf()` to display a single number according to some of the type specifiers listed in Table 10.1.
Notice that we do not only add conversion specifications to the format control string. Any additional text we include will also be printed.

Listing 10.1. Demonstrating Some Type Specifiers

```
<?php
$number = 543;
printf("Decimal: %d<br/>", $number);
printf("Binary: %b<br/>", $number);
printf("Double: %f<br/>", $number);
printf("Octal: %o<br/>", $number);
printf("String: %s<br/>", $number);
printf("Hex (lower): %x<br/>", $number);
printf("Hex (upper): %X<br/>", $number);
?>
```

Put these lines into a text file called `printftest.php` and place this file in your web server document root. When you access this script through your web browser, it should look something like Figure 10.1. As you can see, `printf()` is a quick way of converting data from one number system to another and outputting the result.

Figure 10.1. Demonstrating conversion specifiers.


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/printftest.php
Go

```
Decimal: 543
Binary: 1000011111
Double: 543.000000
Octal: 1037
String: 543
Hex (lower): 21f
Hex (upper): 21F
```

When specifying a color in HTML, you combine three hexadecimal numbers between 00 and FF, representing the values for red, green, and blue. You can use `printf()` to convert three decimal numbers between 0 and 255 to their hexadecimal equivalents:

```
<?php
$red = 204;
$green = 204;
$blue = 204;
printf("#%X%X%X", $red, $green, $blue);
 // prints "#CCCCCC" ?>
```

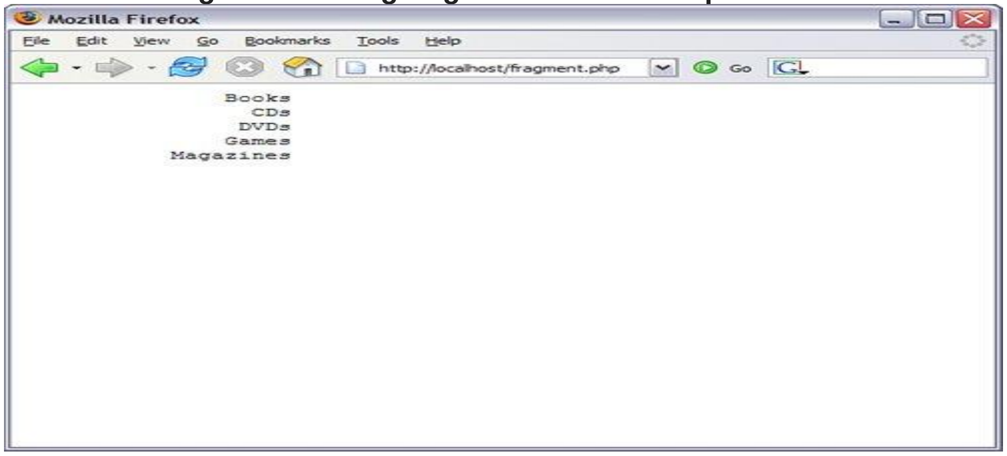
Specifying a Field Width

You can specify the number of spaces within which your output should sit. The field width specifier is an integer that should be placed after the percent sign that begins a conversion specification (assuming that no padding specifier is defined). The following fragment outputs a list of four items, all of which sit within a field of 20 spaces. To make the spaces visible on the browser, we place all our output within a pre element:

```
<?php
echo "<pre>";
printf("%20s\n", "Books");
 printf("%20s\n", "CDs");
printf("%20s\n", "DVDs");
 printf("%20s\n", "Games");
printf("%20s\n", "Magazines");
 echo "</pre>";
?>
```

Figure 10.2 shows the output of this fragment.

Figure 10.2. Aligning with field width specifiers.


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/fragment.php
Go

```
          Books
            CDs
           DVDs
          Games
      Magazines
```

By default, output is right-aligned within the field you specify. You can make it left-aligned by prepending a minus () symbol to the field width specifier:

```
printf("%-20s\n", "Left aligned");
```

Q) Explain about classes and objects in PHP?

We can imagine our universe made of different objects like sun, earth, moon etc. Similarly we can imagine our car made of different objects like wheel, steering, gear etc. Same way there is object oriented programming concepts which assume everything as an object and implement a software using different objects.

Object Oriented Concepts

Before we go in detail, lets define important terms related to Object Oriented Programming.

- **Class** − This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instancesof the same kind (or class) of object.
- **Object** − An individual instance of the data structure defined by a class. You define aclass once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable** − These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function** − These are the function defined inside a class and are used to access object data.
- **Inheritance** − When a class is defined by inheriting existing function of a parent classthen it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class** − A class that is inherited from by another class. This is also called abase class or super class.
- **Child Class** − A class that inherits from another class. This is also called a subclass or derived class.
- **Polymorphism** − This is an object oriented concept where same function can be usedfor different purposes. For example function

name will remain same but it make take different number of arguments and can do different task.

- **Data Abstraction** − Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation** − refers to a concept where we encapsulate all the data and member functions together to form an object.

How to create classes?

In order to create a **class**, we group the code that handles a certain topic into one place. For example, we can group all of the code that handles the users of a blog into one class, all of thecode that is involved with the publication of the posts in the blog into a second class, and all the code that is devoted to comments into a third class.

For the example given below, we are going to create a Car class into which we will group all of the code which has something to do with cars.

```
class Car
{
 // The code
}
```

□ We declare the class with the class keyword.

□ We write the name of the class and capitalize the first letter.

How to add properties to a class?

We call **properties** to the variables inside a class. Properties can accept values like strings, integers, and booleans (true/false values), like any other variable. Let's add some properties to the Car class.

```
class Car {
 public
 $comp;
 public $color =
 'beige'; public
 $hasSunRoof = true;
}
```

☞ We put the public keyword in front of a class property.

☞ The naming convention is to start the property name with a lower case letter.

☞ If the name contains more than one word, all of the words, except for the first word, startwith an upper case letter. For example, $color or $hasSunRoof.

How to create objects from a class?

We can create several objects from the same class, with each object having its own set of properties.
In order to work with a class, we need to create an **object** from it. In order to create an object, we use the new keyword. For example:

```
$bmw = new Car ();
```

☞ We created the object $bmw from the class Car with the new keyword.

- ☞ The process of creating an object is also known as **instantiation**.
- ☞ We can create more than one object from the same class.

```
$bmw = new Car ();

$mercedes = new Car ();
```

How to get an object's properties?

Once we create an object, we can get its properties. For example:

```
echo $bmw ->
color; echo
$mercedes -> color;
```

- ☞ In order to get a property, we write the object name, and then dash greater than (->),and then the property name.
- ☞ Note that the property name does not start with the $ sign; only the object name startswith a $.

Result: **beigebeige**

How to set the object's properties?

In order to set an object property, we use a similar approach.

For example, in order to set the color to 'blue' in the bmw object:

```
$bmw -> color = 'blue';
```

and in order to set the value of the $comp property for both objects:

```
$bmw -> comp = "BMW";
$mercedes -> comp = "Mercedes Benz";
```

Once we set the value of a property, we can get its value.
In order to get the color of the $bmw object, we use the following line of code:

```
echo $bmw -> color;
```

Result: blue

We can also get the company name and the color of the second car object.

```
echo $mercedes ->
color;echo
$mercedes -> comp;
```

Result: beige
Mercedes Benz

How to add methods to a class?

The classes most often contain functions. A function inside a class is called a **method**. Here we add the **method** hello() to the class with the prefix public.

```
class Car {

  public

  $comp;
  public $color =
  'beige'; public
  $hasSunRoof = true;

  public function hello()
  {
    return "beep";
  }
}
```

☞ We put the public keyword in front of a method.

☞ The naming convention is to start the function name with a lower case letter.

We can approach the methods similar to the way that we approach the properties, but we first need to create at least one object from the class.

```
$bmw = new Car ();
$mercedes = new Car ();

echo $bmw ->
hello(); echo
$mercedes -> hello();
```

<u>Result:</u>
beepbeep

Here is the full code

```
<?php
// Declare the class
class Car {
  // properties
  public $comp;
  public $color = 'beige';
  public $hasSunRoof = true;

   // method that
  says hellopublic
  function hello()
  {
    return "beep";
  }
}

// Create an instance
$bmw = new Car ();
$mercedes = new Car ();
```

```
// Get the values
echo $bmw -> color; // beigeecho "<br />";
echo $mercedes -> color; // beigeecho "<hr />";

// Set the values
$bmw -> color = 'blue';
$bmw -> comp = "BMW";
$mercedes -> comp = "Mercedes Benz";

// Get the values again
echo $bmw -> color; // blue echo "<br />";
echo $mercedes -> color;
// beige echo "<br />";
echo $bmw -> comp;
// BMW echo "<br />";
echo $mercedes -> comp;
// Mercedes Benz echo "<hr />";
// Use the methods to get a beep echo $bmw -> hello();
// beep echo "<br />";
echo $mercedes -> hello();
// beep
```

Q) Saving State Between Function Calls with the 'static' Statement ?

If you declare a variable within a function in conjunction with the static statement, the variable remains local to the function, and the function "remembers" the value of the variable from execution to execution.

Example:

```
<?php
  function keep_track()
  {
    STATIC $count = 0;
    $count++;
    print $count;
    print "<br />";
  }

  keep_track();
  keep_track();
  keep_track();
?>
```

More About Arguments

Setting Default Values for Arguments

You can also create functions with optional parameters — just insert the parameter name, followed by an equals (=) sign, followed by a default value, like this.

Example:

```
<?php
// Defining function
function customFont($font, $size=1.5)
{
   echo "<p style=\"font-family: $font; font-size: {$size}em;\">Hello,
world!</p>";
}

// Calling function
customFont("Arial", 2);
customFont("Times", 3);
customFont("Courier");
?>
```

Q)Explain Code Blocks And Browser Output?

There are two techniques.

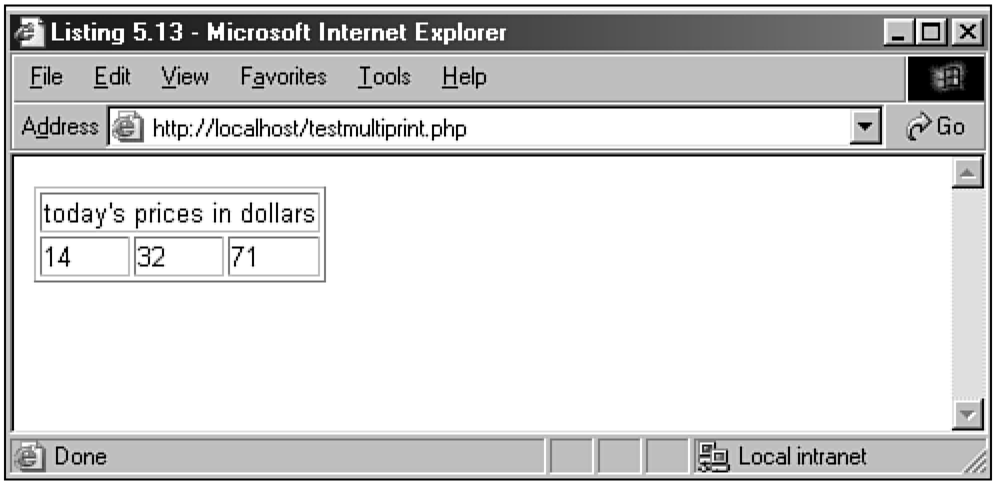
Imagine a script that outputs a table of values only when a variable is set to the Boolean value true. Listing 5.13 shows a simplified HTML table constructed with the code block of an if statement.

A Code Block Containing Multiple print() Statements

```
 <html>
<body>
<?php
$display_prices = true;
if ( $display_prices ) {
   print "<table border=\"1\">";
   print "<tr><td colspan=\"3\">";
   print "today's prices in dollars";
   print "</td></tr>";
   print "<tr><td>14</td><td>32</td><td>71</td></tr>";
   print "</table>";
 }
 ?>
</body>
</html>
```

If $display_prices is set to true in line 7, the table is printed. For the sake of readability, we split the output into multiple print() statements, and once again escape any quotation marks.

Put these lines into a text file called testmultiprint.php, and place this file in your Web server document root. When you access this script through your Web browser, it should look like Figure


Listing 5.13 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/testmultiprint.php
Go
Done
Local intranet

today's prices in dollars		
14	32	71

There's nothing wrong with the way this is coded, but we can save ourselves some typing by simply slipping back into HTML mode within the code block.

```
<html>
<body>
<?php
$display_prices = true;
if ( $display_prices )
{
 ?>
<table border="1">
<tr><td colspan="3">today's prices in dollars</td></tr>
<tr><td>14</td><td>32</td><td>71</td>
</table>
<?php
}
?>
</body>
</html>
```

The important thing to note here is that the shift to HTML mode on line 9 only occurs if the condition of the if statement is fulfilled. This can save us the bother of escaping quotation marks and wrapping our

output in print() statements. It might, however, affect the readability of our code in the long run, especially as our script grows larger.

Q)Explain about date and time functions.

The date/time functions allow you to get the date and time from the server where your PHP script runs. You can then use the date/time functions to format the date and time in several ways.

The PHP Date() Function:-

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

The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the Unix time format.

Ex:-

```
<?php
$today = date("d/m/Y");
echo $today;
?>
```

Formatting the Dates and Times with PHP:-

Here are some the date-related formatting characters that are commonly used in format string:

- ☞ **d - Represent day of the month; two digits with leading zeros (01 or 31)**
- ☞ **D - Represent day of the week in text as an abbreviation (Mon to Sun)**
- ☞ **m - Represent month in numbers with leading zeros (01 or 12)**
- ☞ **M - Represent month in text, abbreviated (Jan to Dec)**
- ☞ **y - Represent year in two digits (08 or 14)**
- ☞ **Y - Represent year in four digits (2008 or 2014)**

Ex:-

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

The PHP time() Function:-

The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1 1970 00:00:00 GMT).

Ex:-

```
<?php
// Executed at March 05, 2014 07:19:18
$timestamp = time();
```

```
echo($timestamp);
?>
```

The above example produce the following output.

```
        1394003958
```

We can convert this timestamp to a human readable date through passing it to the previously introduce date() function.

Example:-

```
<?php
$timestamp = 1394003958;
echo(date("F d, Y h:i:s", $timestamp));
?>
```

The PHP mktime() Function:-

The mktime() function is used to create the timestamp based on a specific date and time. If no date and time is provided, the timestamp for the current date and time is returned.

The syntax of the mktime() function can be given with:

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

The following example displays the timestamp corresponding to 3:20:12 pm on May 10, 2014:

Example:-

```
<?php
// Create the timestamp for a particular
date
echo mktime(15, 20, 12, 5, 10, 2014);
?>
```

The PHPStrtotime() function:-

strtotime — passing any English textual datetime description into a Unix timestamp.

Example :-

```
<?php
echo strtotime("now"), "\n";
echo strtotime("10 September 2000"), "\n";
echo strtotime("+1 day"), "\n";
echo strtotime("+1 week"), "\n";
echo strtotime("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime("next Thursday"), "\n";
echo strtotime("last Monday"), "\n";
?>
```

The PHP date_format() function:-

The date_format() function returns a date formatted according to the specified format.

```
<?php
$date=date_create("2013-03-15");
echo date_format($date,"Y/m/d H:i:s");
?>
```

The PHP sunset() and sunrise() functions:-

The date_sunrise() function returns the sunrise time for a specified day and location.

The date_sunset() function to return the sunset time for a specified day and location.

Ex:-

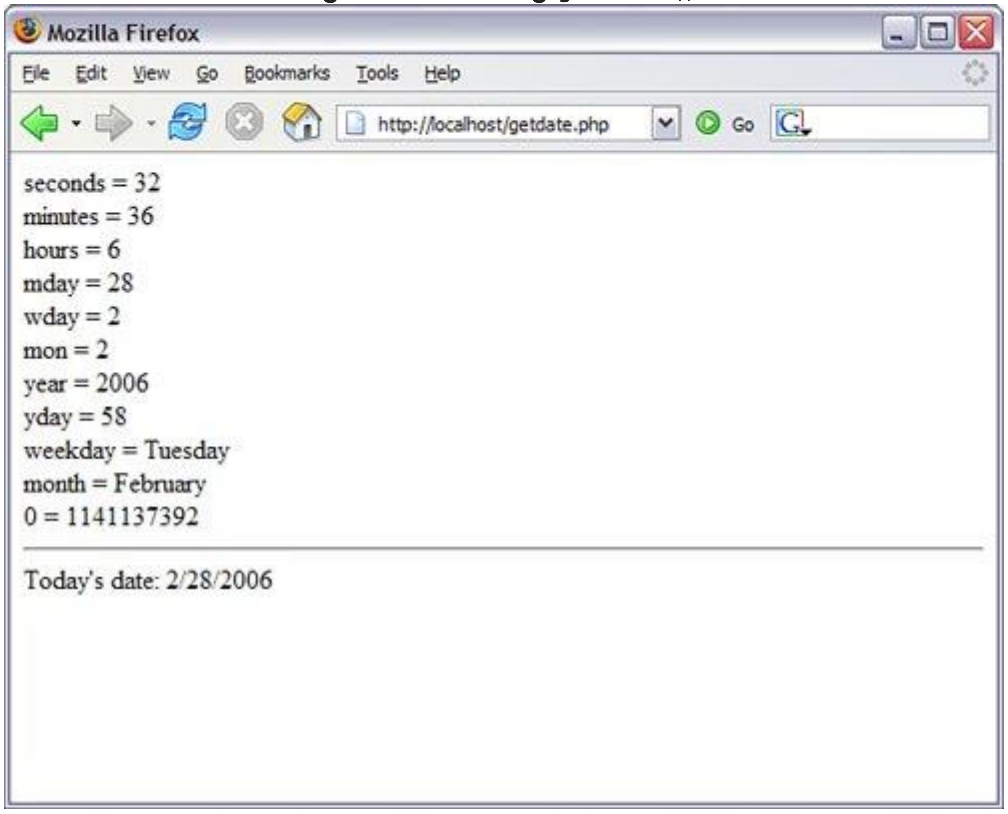
```
<?php
echo("Date: " . date("D M d Y"));
echo("<br>Sunrise time: ");
echo(date_sunrise(time()));
echo("<br>Sunset time: ");
echo(date_sunset(time()));
?>
```

lists the elements contained in the array returned by `geTDate()`.

Table 10.3. The Associative Array Returned by getdate()

Key	Description	Example
`Seconds`	Seconds past the minute (059)	`43`
`Minutes`	Minutes past the hour (059)	`30`
`Hours`	Hours of the day (023)	`8`
`Mday`	Day of the month (131)	`9`
`Wday`	Day of the week (06)	`1`
`mon`	Month of the year (112)	`8`
`year`	Year (4 digits)	`2004`
`yday`	Day of the year (0365)	`221`
`weekday`	Day of the week (name)	`Monday`
`month`	Month of the year (name)	`August`
`0`	Time stamp	`1092065443`

Figure 10.5. Using `getdate()`.


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/getdate.php
Go

```
seconds = 32
minutes = 36
hours = 6
mday = 28
wday = 2
mon = 2
year = 2006
yday = 58
weekday = Tuesday
month = February
0 = 1141137392

Today's date: 2/28/2006
```

Listing 10.4. Acquiring Date Information with getdate()

```
<?php
$date_array = getdate(); // no argument passed so today's
date will be used
foreach ($date_array as $key => $val) {
    echo "$key = $val<br>";
}
?>
<hr/>
<?php
echo "<p>Today's date:
".$date_array['mon']."/".$date_array['mday']."/".
    $date_array['year']."</p>";
?>
```

. Some Format Codes for Use with date()

Format	Description	Example
A	am or pm (lowercase)	am
A	AM or PM (uppercase)	AM
D	Day of month (number with leading zeroes)	28
D	Day of week (three letters)	Tue
E	Timezone identifier	America/Los_Angeles
F	Month name	February
H	Hour (12-hour formatleading zeroes)	06
H	Hour (24-hour formatleading zeroes)	06
G	Hour (12-hour formatno leading zeroes)	6
G	Hour (24-hour formatno leading zeroes)	6
I	Minutes	45
J	Day of the month (no leading zeroes)	28
L	Day of the week (name)	Tuesday
L	Leap year (1 for yes, 0 for no)	0
M	Month of year (numberleading zeroes)	2
M	Month of year (three letters)	Feb
N	Month of year (numberno leading zeroes)	2
S	Seconds of hour	26
S	Ordinal suffix for the day of the month	th
R	Full date standardized to RFC 822 (http://www.faqs.org/rfcs/rfc822.html)	Tue, 28 Feb 2006 06:45:26 -0800
U	Time stamp	1141137926
Y	Year (two digits)	06
Y	Year (four digits)	2006
Z	Day of year (0365)	28
Z	Offset in seconds from GMT	-28800

UNIT III

1. Working with Forms:

Creating Forms, Accessing Form Input with User defined Arrays, Combining HTML and PHP code on a single Page, Using Hidden Fields to save state, Redirecting the user, Sending Mail on Form Submission, Working with File Uploads.

2.Working with Cookies and User Sessions: Introducing Cookies, Setting a Cookie with PHP, Session Function Overview, Starting a Session, Working with session variables, passing session IDs in the Query String, Destroying Sessions and Unsetting Variables, Using Sessions in an Environment with Registered Users.

3.Working with Files and Directories: Including Files with include(), Validating Files, Creating and Deleting Files, Opening a File for Writing, Reading or Appending, Reading from Files, Writing or Appending to a File, Working with Directories, Open Pipes to and from Process Using popen(), Running Commands with exec(), Running Commands with system() or passthru().

4.Working with Images: Understanding the Image-Creation Process, Necessary Modifications to PHP, Drawing a New Image, Getting Fancy with Pie Charts, Modifying Existing Images, Image Creation from User Input.

FORMS IN PHP

A Document that containing black fields, that the user can fill the data or user can select the data. Casually the data will store in the data base.

Get and Post Methods in PHP

PHP provides two methods through which a client (browser) can send information to the server. These methods are given below, and discussed in detail:

1. **GET method**
2. **POST method**

Get and Post methods are the HTTP request methods used inside the <*form*> tag to send form data to the server.

HTTP protocol enables the communication between the client and the server where a browser can be the client, and an application running on a computer system that hosts your website can be the server.

GET method

The GET method is used to submit the HTML FORM data. This data is collected by the predefined $_GET variable for processing.

POST method

Similar to the GET method, the POST method is also used to submit the HTML form data. But the data submitted by this method is collected by the predefined super global variable $_POST instead of $_GET.

Note that the "post" method is more secure than the "get" method because the data sent using the POST method is not visible to user.

$_REQUEST variable

The $_REQUEST variable is a superglobal variable, which can hold the content of both $_GET and $_POST variable. In other words, the PHP $_REQUEST variable is used to collect the form data sent by either GET or POST methods. It can also collect the data for $_COOKIE variable because it is not a method-specific variable.

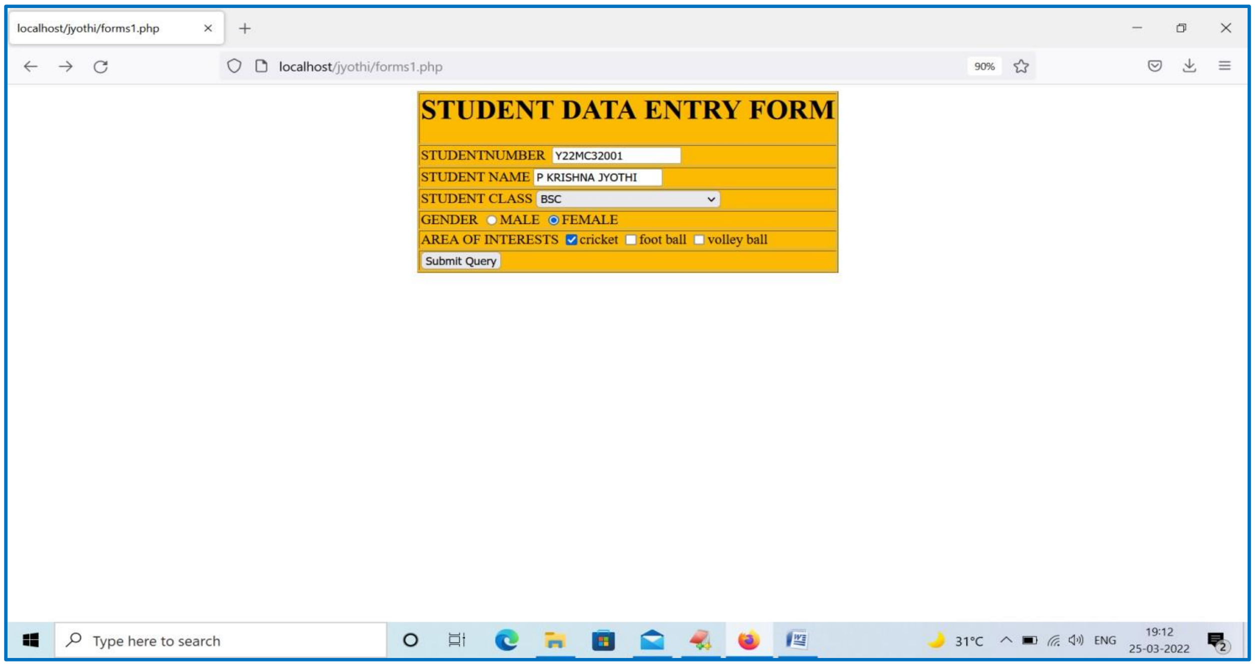
EXAMPLE FOR A FORM TO CREATE STUDENT DETAILS

```
<form name="forms1.php" method="POST" action="forms2.php">
<table border="" bgcolor="#fcba03" align="center">
<td>
<h1>STUDENT DATA ENTRY FORM</h1>
</td>
<tr>
<td>STUDENTNUMBER
<input type="text" value="enter your regd no" name="sno"><tr>
<td>STUDENT NAME
<input type="text" value="enter your NAME" name="sname"><tr>
<td>STUDENT CLASS
<select name="studentclass">
        <option value="0" selected >SELECT ANY ONE SECTION
        <option value="BSC">BSC
        <option value="BCA">BCA
        <option value="MCA">MCA
        </select>

<tr>
<td>GENDER
<input type="radio" name="gender" value="M">MALE
<input type="radio" name="gender" value="F">FEMALE
<tr>
<td>AREA OF INTERESTS
<input type="checkbox" name="interests[]" value="cricket">cricket
<input type="checkbox" name="interests[]" value="football">foot ball
<input type="checkbox" name="interests[]" value="volleyball">volley ball
<tr>
<td><input type="submit">

</td></table>
</form>
```

After that create forms2.php

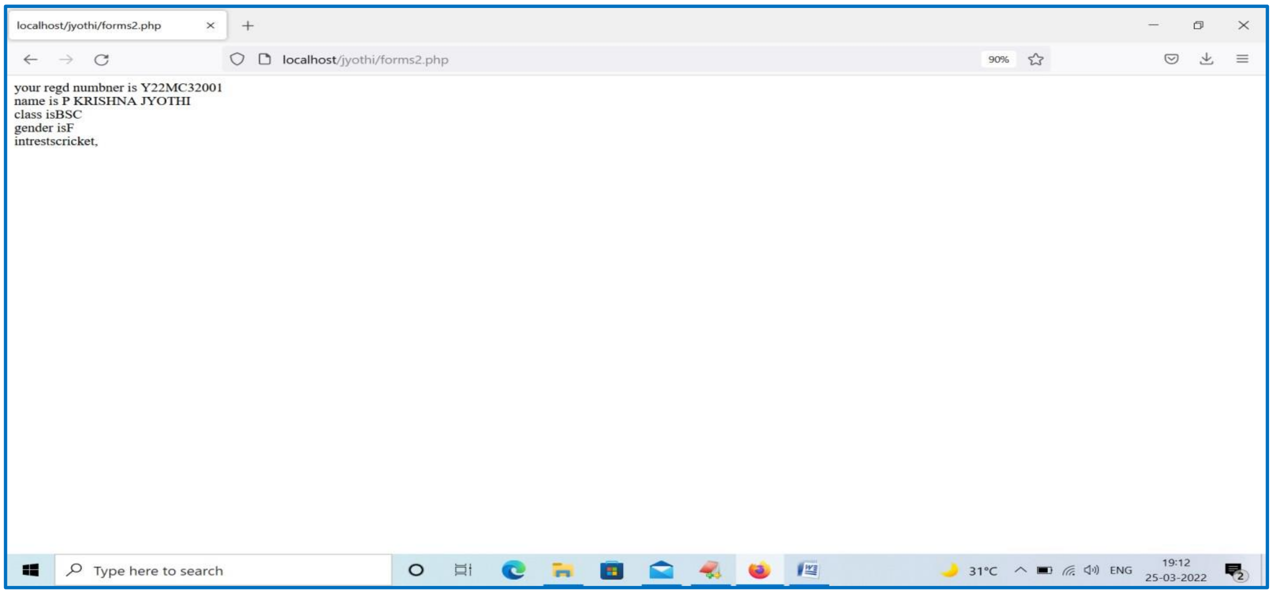
```
<?php
$stnum=$_POST['sno'];
echo "your regd numbner is ".$stnum;
$sname=$_POST['sn
ame'];echo     "<br>";
echo     "name     is
".$sname;
$sclass=$_POST['studentc
lass'];
echo  "<br>";  echo  "class
is".$sclass;
$gen=$_POST['g
ender'];
echo "<br>";echo"gender is".$gen;
$intr=$_POST['interests'];echo "<br>";
foreach($intr as $chk1)
  {
   echo"intrests".$chk1.",";

  }?>
```

When we run the forms1.php the BELOW out put is formed



STUDENT DATA ENTRY FORM

STUDENTNUMBER	Y22MC32001
STUDENT NAME	P KRISHNA JYOTHI
STUDENT CLASS	BSC
GENDER	○ MALE [✓ FEMALE]
AREA OF INTERESTS	[✓ cricket] ☐ foot ball ☐ volley ball
Submit Query	



```
your regd numbner is Y22MC32001
name is P KRISHNA JYOTHI
class isBSC
gender isF
intrestscricket,
```

Q) Using Hidden Fields to save state

The script has no way of knowing how many guesses a user has made. We can use a hidden field to keep track of this. A hidden field behaves exactly the same as a text field, except that the user cannot see it, unless he views the HTML source of the document that contains it.

Create a form hidden2.php

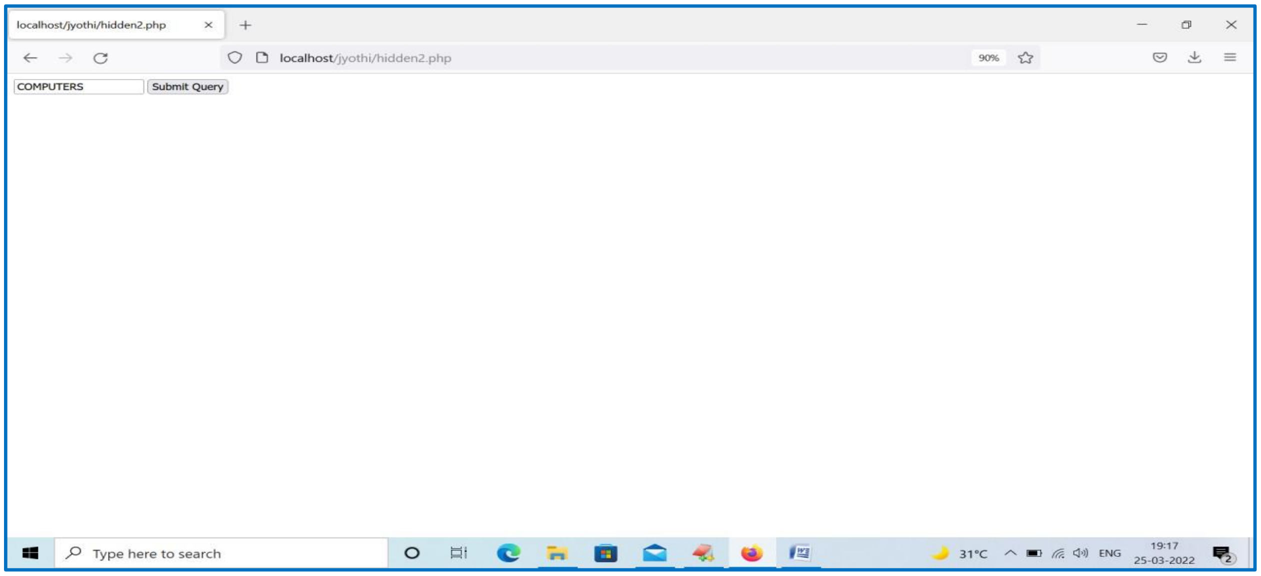
```
<form name="hidden2.php" action="hidden3.php" method="POST">
<td><input type="hidden" name="snum" value="<?echo 'Y22MC32001';
```

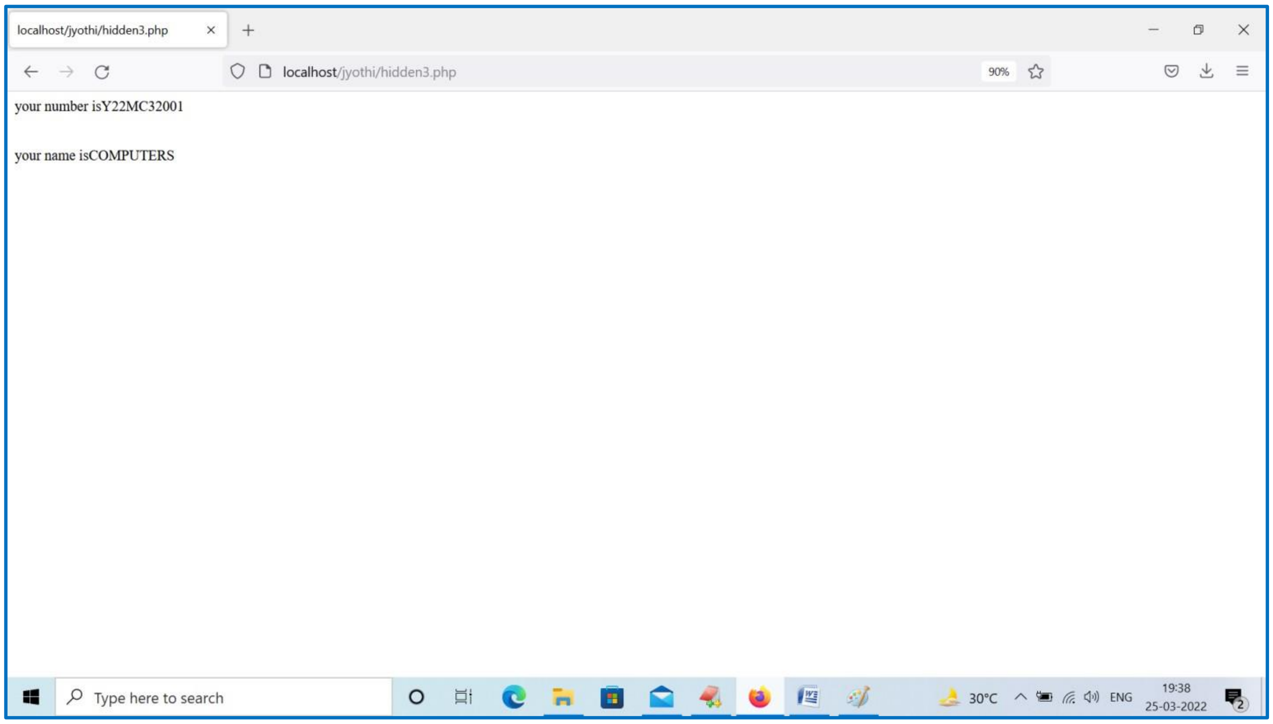
```
?>"></td>

<td><input type="text" name="stname" value="<?echo  'COMPUTERS';
?>"></td>

<td><input type="submit"></td>
```

Create hidden3.php

```
<? php
$a=$_POST['snum'];
$b=$_POST['stname'];
echo "your number is".$a;echo "<br>";
echo "<br>";
echo "your name is".$b;
?>
```

When we execute the hidden2.php





localhost/jyothi/hidden3.php

```
your number isY22MC32001

your name isCOMPUTERS
```

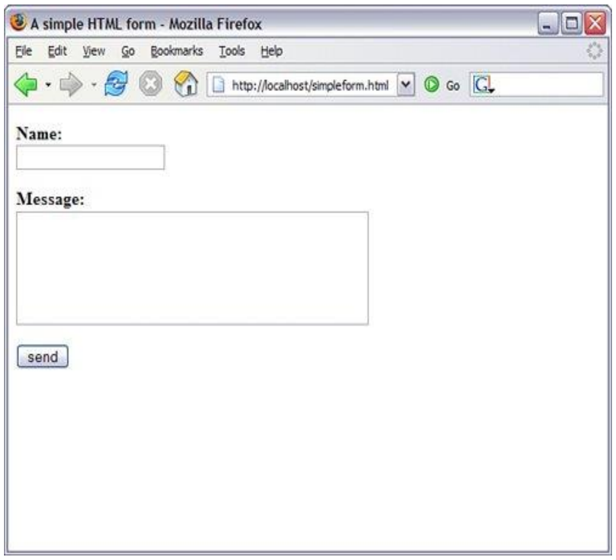
Q) Accessing Form Input with User-Defined Arrays

```
<html>
<head>
<title>A simple HTML form</title>
</head>
<body>
<form action="send_simpleform.php" method="POST">
<p><strong>Name:</strong><br/>
<input type="text" name="user"/></p>
<p><strong>Message:</strong><br/>
<textarea name="message" rows="5" cols="40"></textarea></p>
<p><input type="submit" value="send"/></p>
</form>
</body>
</html>
```

Reading Input from a Form

```
 <?php
 echo "<p>Welcome <b>".$_POST["user"]."</b>!</p>";
 echo "<p>Your message
is:<br/><b>".$_POST["message"]."</b></p>";
?>
```

The form created by simpleform.html.


A simple HTML form - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/simpleform.html
Go

Name:

Message:

send

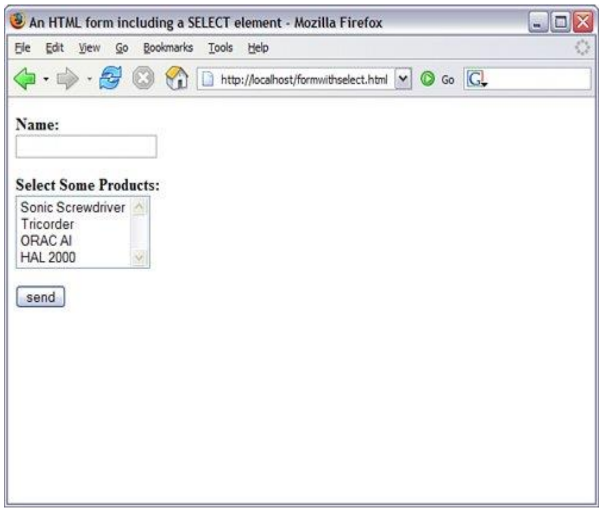
```
<html>
<head>
<title>An HTML form including a SELECT element</title>
</head>
<body>
<form action="send_formwithselect.php" method="POST">
<p><strong>Name:</strong><br/>
<input type="text" name="user"/>
<p><strong>Select Some Products:</strong><br/>
<select name="products[]" multiple="multiple">
<option value="Sonic Screwdriver">Sonic Screwdriver</option>
<option value="Tricoder">Tricorder</option>
<option value="ORAC AI">ORAC AI</option>
<option value="HAL 2000">HAL 2000</option>
</select>
<p><input type="submit" value="send"/></p>
</form>
</body>
</html>
```

Reading Input from the Form in Listing 11.3

```
 <?php
 echo "<p>Welcome <b>".$_POST["user"]."</b>!</p>";
echo "<p>Your product choices are:<br/>";
if (!empty($_POST["products"]))
{
echo "<ul>";
foreach ($_POST["products"] as $value) {
echo "<li>$value</li>";   8:     }
echo "</ul>"; 10: } 11: ?>
```

the form created in Listing 11.3

.


An HTML form including a SELECT element - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/formwithselect.html
Go

Name:

Select Some Products:

Sonic Screwdriver
Tricorder
ORAC AI
HAL 2000

send

Q) Sending mail on form submission

Before sending email , we must make sure whether the system is properly configured

System configuration for the mail ( ) function :

We can use the mail ( ) function to send mail, a few directives

must be set up in php. ini file so that the function works

properly.

php.ini should contain the following lines

[mail function]; for win 32 only SMTP = localhost ; for win 32 only

sendmail_from = me@ localhost.com; for unix only.

Creating the form : **create a form and name it as feedback.**

php.

The form has an action sendmail.php.

```
<html>
<head>
<title> email form </ title>
</head >
<body>
<form  action = " sendmail.php"  method = "post">
<p> name :<br> <input type = "text" size ="25" name =
"name"> </p>
```

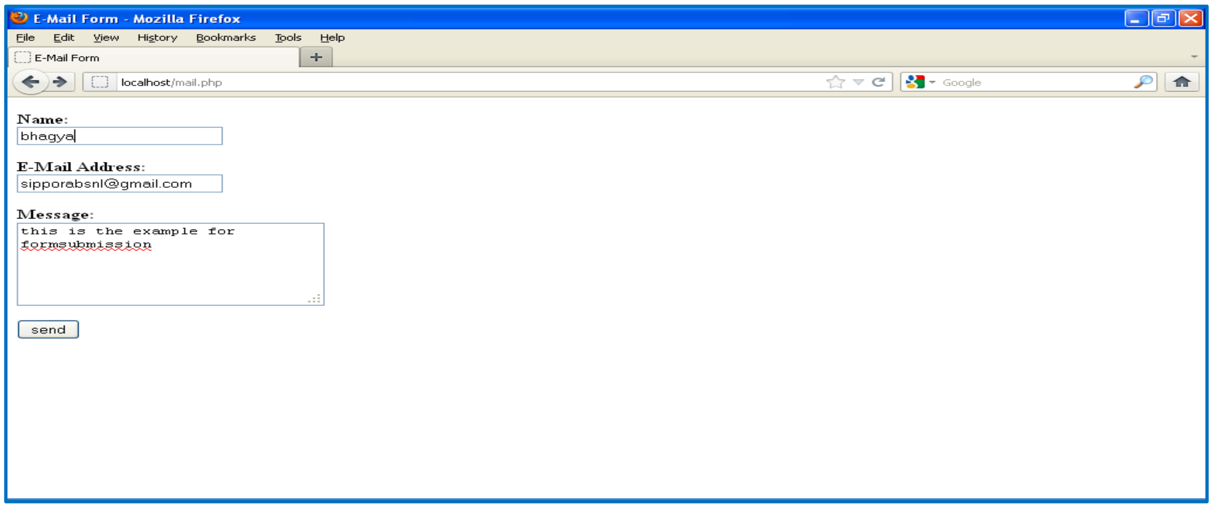
```
<p> e-mail addres :<br>
<input type = " text" size = "25" name= "email"> </p>
<p> message :<br>
<textarea name = "message" cols = "30" rows= "5"></text
area > </p >
< p > < input type = " submit" value = "send" > < /p >
< /form >
< /body >
< / html >
```

Now, create a script that sends this form to recipient

Creating script to send mail :

```
< html >
< head >
< title > sending mail < / title >
</ head >
< body >
<? php
echo " < p > thankyou";
 $ _post [" name"]. for ur message </p>" ;
echo " < p > your mail address is" .$_post ["email" ] </p>" ;
echo " < p > your message was : < br >" ;
echo $_post [ " message " ] ." < /p>";
$msg = "name:".$_POST [" name" ] . " \n";
$msg . = "e-mail:". $_POST ["email"]."\n";
$msg. = " message :". $_POST [" message"]. "\n" ;
$recipient = "you @yourdomain .com" ;
$subject = " form submission results" ;
$mail headers = " from : mywebsite < default address @ your domain .
com > \n" ;
$mail headers .= " reply – to" . $_POST ["email"] ; mail ( $recipient ,
$subject ,$msg , $mail headers ) ;
? >
</body >
< /html >
```

In the above example mail( ) function requires four parameters , the recipient , the subject , the message and any additional mail headers put the above code in a file send mail.php and run it to get the output.


E-Mail Form - Mozilla Firefox
File Edit View History Bookmarks Tools Help
E-Mail Form
localhost/mail.php
Google

Name:
bhagya

E-Mail Address:
sipporabsnl@gmail.com

Message:
this is the example for formsubmission

send

Q) Working with File Uploads

A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script

First we need to create the HTML .HTML forms that include file upload fields must include an ENCTYPE ARGUMENT:

ENCTYPE= ”mulitipart/form-data”

create form with name imageupload.php

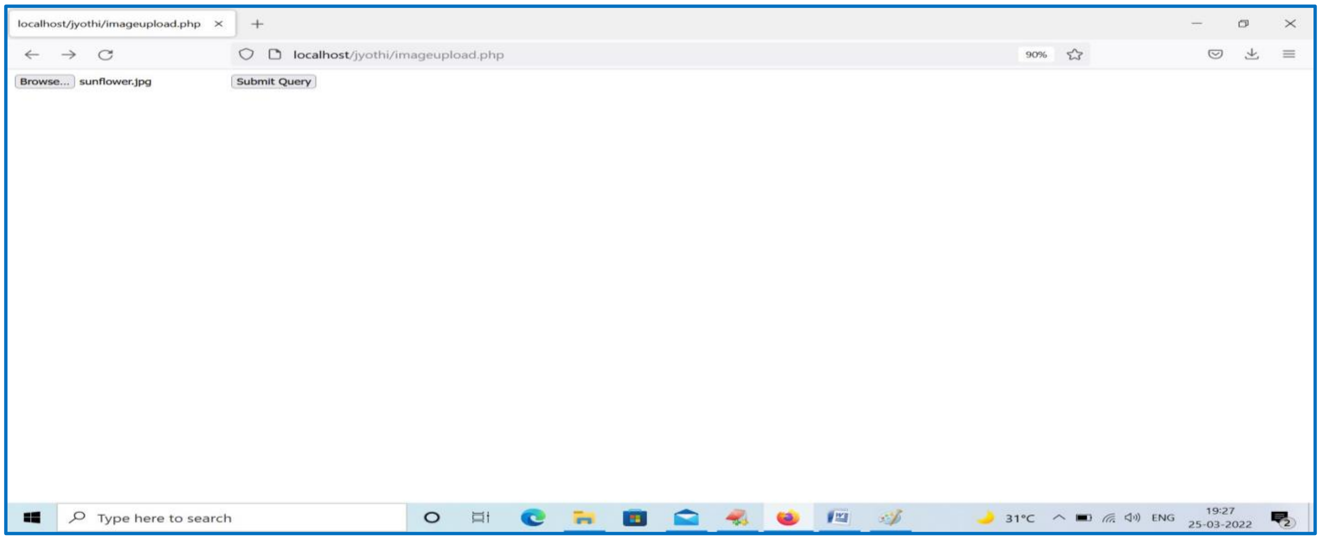
```
<form method="POST" action="imageupload2.php"
enctype="multipart/form-data">
<input type="file" name="uploadfile" value="">
<input type="submit" name="upload">
</form>
```

Create a form with name imageupload2.php

```
<?php
$filename = $_FILES["uploadfile"]["name"];
$tempname = $_FILES["uploadfile"]["tmp_name"];
$folder = "images/".$filename;
if
(move_uploaded_file($tem
pname, $folder)) {echo
$msg = "Image uploaded
successfully";
}
```

```
else
{
echo $msg = "Failed to upload image";
}
?>
```

When we execute imageupload.php


localhost/jyothi/imageupload.php
localhost/jyothi/imageupload.php
Browse... sunflower.jpg
Submit Query

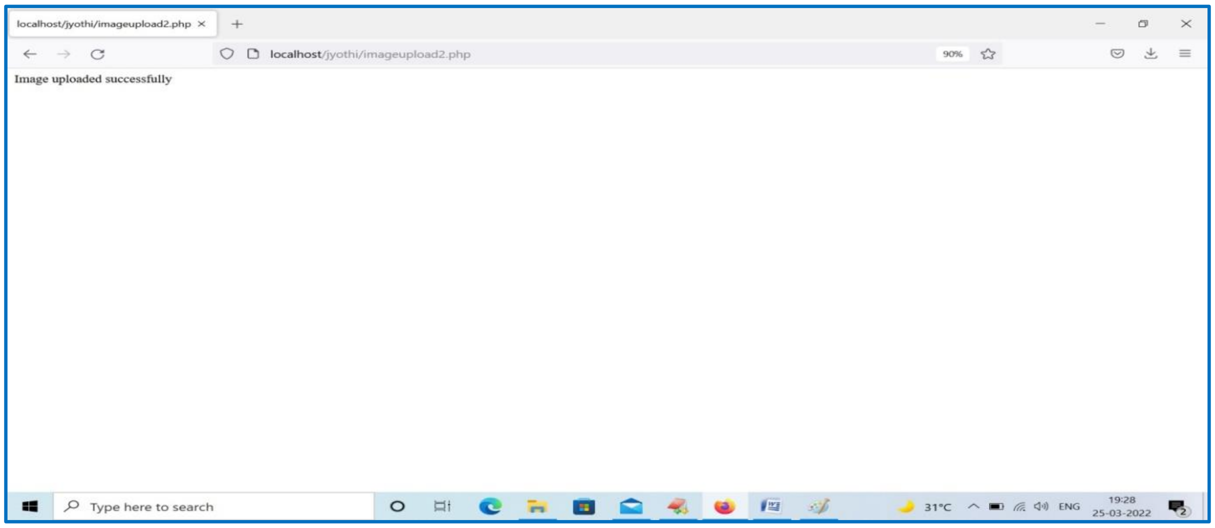

localhost/jyothi/imageupload2.php
localhost/jyothi/imageupload2.php

Image uploaded successfully

Working With Files and Directories

Topics:

- ☞ **Include statement**
- ☞ **Validating files**
- ☞ **Getting date information about file**
- ☞ **Opening files**
- ☞ **Reading from files**
- ☞ **Writing or appending to files**
- ☞ **Locking files**
- ☞ **Working with directories**
- ☞ **Running unix commands with php functions**

Including files with include ()

The include() statement enables you to incorporate other files into your PHP document. The include() requires a single argument, a relative path to the file to be included.
The following example creates a simple PHP script that uses include() to incorporate and output contents of file.

```
<?php
include (“myinclude.php”);
?>
```

The following content is typed in myinclude.php

```
<? php
I have been included.
?>
```

If we run the above PHP code the following output is displayed. I have been included

Returning a value from included document:-

- **Included files in PHP can return a value in the same way as functions do.**

Example:

```
<?php
  $res=include  (“returnvalue.php”);
 echo “the included file returned”. $res;
 ?>
```

An include file that returns a value returnvalue.php

```
<? Php
$r=(4+4);
return $r;
?>
```

Output:- the included file returned 8.

Validating Files:-

- ✓ checking a file for existence with file _exists ().
- ✓ we can test for the existence of a file with file_exists() function.
- ✓ The function requires file name as its argument. If file is found the function returns true otherwise false.

```
if (file_exists("test.txt"))
{
 echo "the file exists";

}
```

Checking A File Or Directory:-

- ✓ you can confirm that the entity you are testing is a file using is_file() function.
- ✓ This is_file() function requires file path and returns boolean value.

```
if(is_file("test.txt"))
{
echo "test.txt is a file!";
}
```

We can check that the entity we are testing is a directory using is_dir()

function. It requires path as a argument and returns a Boolean value.

```
if(is_dir("/tmp"))
{
echo "/tmp is a directiory";
}
```

Checking status of file:-

Generally we want to read, write or execute a file.PHP helps you to determine whether we can perform these operations by providing various functions.

The is_readable() function tells you whether you can read a file.

It accepts file path as argument and returns a Boolean value.

```
if(is_readable("test.txt"))

{

 echo "test.txt is readable";

}
```

The is_writable() function tells whether you have proper permission to write a file.This function also accepts file path and returns a Boolean vale.

```
if(is_writable("test.txt"))

{

  echo "this file is writable";

}
```

The is_executable() function tells you whether you can execute the file. The function accepts file path and returns a boolean value.

Q) Getting Date information about file

Some times we need to know when a file was last written or accessed.

PHP provides several functions to provide this information.

We can get the last accessed time of a file using fileatime() function.

- ✓ This function requires file path and returns date of file in which it was last accessed.
- ✓ The returned value is a timestamp.
- ✓ We use date() function to translate the value into human readable form.

Ex:

```
$atime=fileatime("test.txt");
echo "test.txt was last accesed on".date("D d m y",$atime);
```

We can discover modification date of a file with filemtime() function.

It returns date in UNIX epoch format.

Ex:-

```
$t=filemtime("test.txt");
 echo "test.txt was last accesed on".date("D d m y",$t);
```

PHP enables you to test the change time of a document with filectime() function.

Ex:-
```
$ctime=filectime("test.txt");
echo "test.txt was last accesed on".date("D d m y",$ctime);
```

Q) Operations on Files

Creating And Deleting Files:-

If a file does not exist,we can create it with touch() function.
Given a string representing filepath ,touch() creates an empty file of that name.
If the file already exists its contents are not disturbed but modification date will be updated.

```
touch("myfile.txt");
```

we can remove an existing file with unlink() function.unlink() function also accepts file path.

```
unlink("myfile.txt");
```

Opening a File for Writing,Reading or Appending:-

Before we work with files ,we first open it for reading or writing or to perform both tasks.

PHP provides fopen() function for doing so.fopen() function requires a string that contains mode in which file is to be opened.

The most common modes are read(r),write(w) and append(a).

The fopen() function requires a file resource so that we can use later.

To open a file for reading we use

```
$fp=fopen("test.txt","r");
```

To open a file for writing

```
$fp=fopen("test.txt","w");
```

To open a file for appending

```
$fp=fopen("test.txt","a");
```

fopen() function returns false if file cannot be opened for some reason.
After opening and working with a file, you should close it by using the function fclose(). This function requires file resource as an argument which is returned by fopen().

```
fclose($fp);
```

Reading from files:-

PHP provides number of functions to read data from files.These functions enable you to read byte by byte,by whole line and even by single character.

Reading Lines from a file with fgets() and feof():-

To read a line from open file we can use fgets() function which requires file resource returned from fopen()as its argument.

We must also pass fgets() an integer as second argument,which specifies number of bytes that function should read if it doesn't encounter a line end or end of file.

The feof() function does this by returning true when end of file been reached and false otherwise.

The feof() function requires file resource as its argument.

Ex:-opening and reading a file line by line.

```
<?php

$filename=”test.txt”;

$fp=fopen($filename,”r”)                or

die(“couldn’t     open     $filename”);

while(!feof($fp)) {

$line=
fgets($
fp,102
4);
echo
$line.
”<br>”
;
}
?>
```

Reading characters from a file with fget():-

The fgetc() function reads and returns a single character from a file every time it is called. Because a character is always one byte in size, fgetc() does not require a length argument.

Ex:-

```
<?php
$filename=”test.txt”;
$fp=fopen($filename,”r”) or die(“could not open file”);
while(!feof($fp))
{
$char=fgets($fp); echo $char.”<br>”;
}
?>
```

Writing or Appending to file:-

The process for writing and appending to a file are same, the difference lies in the mode you call fopen() function.

When you write to a file, you use the mode argument “w” ,when you call fopen().

```
$fp=fopen(“test.txt”,”w”);
```

When we append to file ,use the mode “a” in fopen().

```
$fp=fopen(“test.txt”,”a”);
```

Writing to a file with fwrite() or fputs():-

The function accepts file resource and a string and then returns writes to file. The fputs() function works exactly in same way.

```
fwrite($fp,”hello world”);
```

```
fputs($fp,”hello world”);
```

ex:-

```
<?php
$filename=”test.txt”;
$fp=fopen($filename) or die(“could not open”);
fwrite($fp,”hello world”);
fclose($fp);
```

```
echo "Appending.........";
$fp=fopen($filename,"a") or die("could not open");
fputs($fp,"and another thing\n");
fclose($fp);
?>
```

Locking files with flock():-

The flock() function locks a file to warn other processes against writing to or reading from that file while current process is working with it.The flock() function requires a valid file resource from an open file and an integer representing the kind of lock you want to set.

Constant	Integer	Locktype	Description
LOCK_SH	1	Shared	Allow other processes to read the file not writing.
LOCK_EX	2	Exclusive	Prevents other processes for reading and writing
LOCK_UN	3	Release	Release a shared or exclusive lock.

We should call flock () directly after calling fopen() and then call it again to release lock before closing file.

```
<?
$fp=fopen("test.txt","a") or die("could not open");
flock($fp,LOCK_EX);//write to file flock($fp,LOCK_UN);
fclose($fp);
?>
```

Q) Working with Directories

PHP provides many functions to work with directories.

Creating Directories with mkdir():-

The mkdir() function enables you create a directory.

The mkdir() function requires a string that represents the path to the directory you want to create and an octal number integer that represents the mode you want to set for the directory.

Ex:-mkdir("testdir",0777);

mkdir("testdir",07555);

Removing directory with rmdir():-

The rmdir() function enables you to remove a directory from file system, if the directory is empty. The rmdir() function requires only a string representing path to directory you want to delete.

```
rmdir("testdir");
```

Opening a directory for Reading with opendir():-

We can open a directory using opendir() function.
The opendir()function requires a string that represents path to directory you want to open.

```
$dh=opendir("testdir");
```

Here $dh is the directory handle.

Reading contents of directory with readdir():-

We can use readdir() to read a file or directory name from a directory. The readdir() function requires a directory handle and returns a string containing the item name.If the end of directory is reached, readdir() returns false.

Ex:-

```
<?php
$dirname=”vig”;
$dh=opendir($dirname) or die(“can not open”);
while(!($file=readdir($dh)==false))
{
     if(is_dir(“dirname/$file”))
     {
        echo “Dir”;
     }
    else
       echo $file.”<br>”;
}
closedir($dh);
?>
```

Running unix commands with php Running

Commands with exec():-

The exec() function is one of several functions we can use to pass commands
to shell. The below example uses exec() function to produce a directory listing with shell based command.

```
<?php
exec("ls",$out-array,$res);
echo "Returned:".$res."<br>";
foreach($out_array as $o)
{
echo $o."<br>";
}
?>
```

Running command with system() or passthru():-

The system() function differs from exec() in that it outputs information directly to the browser without programmatic intervention.

The passthru() function follows the syntax of system() function but it behaves differently.When using passthru() any output from shell command is not buffered on its way back to you.

Working with Cookies and User Sessions

Topics:-

- ✓ Creating and Deleting Cookie
- ✓ Session Variables and their Work
- ✓ Starting and Resuming a Session
- ✓ Storing Variables in a Session
- ✓ Destroying Sessions
- ✓ Unsetting Session Variables

Creating Cookie with PHP

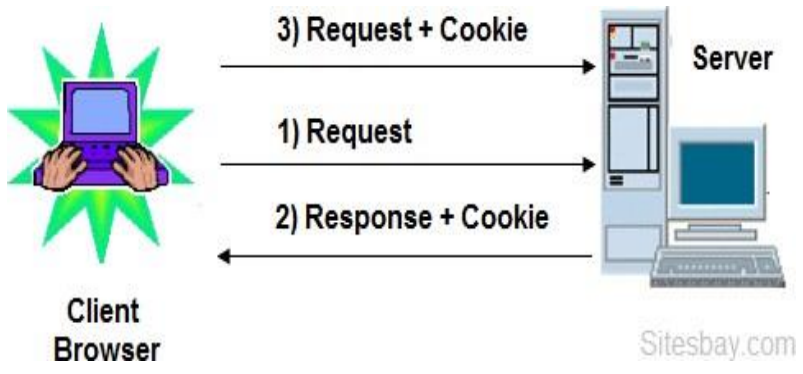
What is a Cookie?

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

(OR)

A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information such as

username that the site can retrieve to personalize the page when user visit the website next time.



Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

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

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

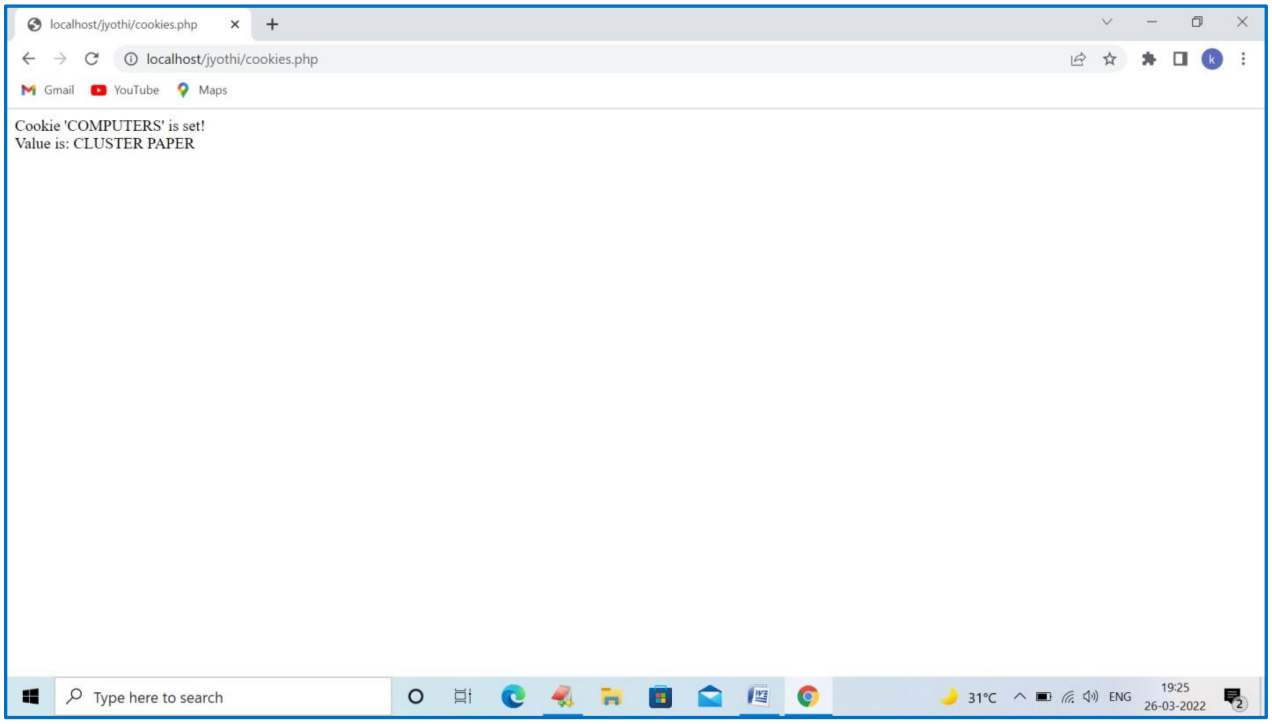
The parameters of the setcookie() function have the following meanings:

Parameter	Description
name	The name of the cookie.
value	The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.
expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
path	Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.
domain	Specify the domain for which the cookie is available to e.g www.example.com.
secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.

Example : cookies.php

```
<?php
$cookie_name = "COMPUTERS";
$cookie_value = "CLUSTER PAPER";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); //
86400 = 1 day
?>
<html>
<body>
<?php if(!isset($_COOKIE[$cookie_name]))
{
 echo "Cookie named '" . $cookie_name . "' is not set!";
}
else
{
 echo "Cookie '" . $cookie_name . "' is set!<br>"; echo
 "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Output:

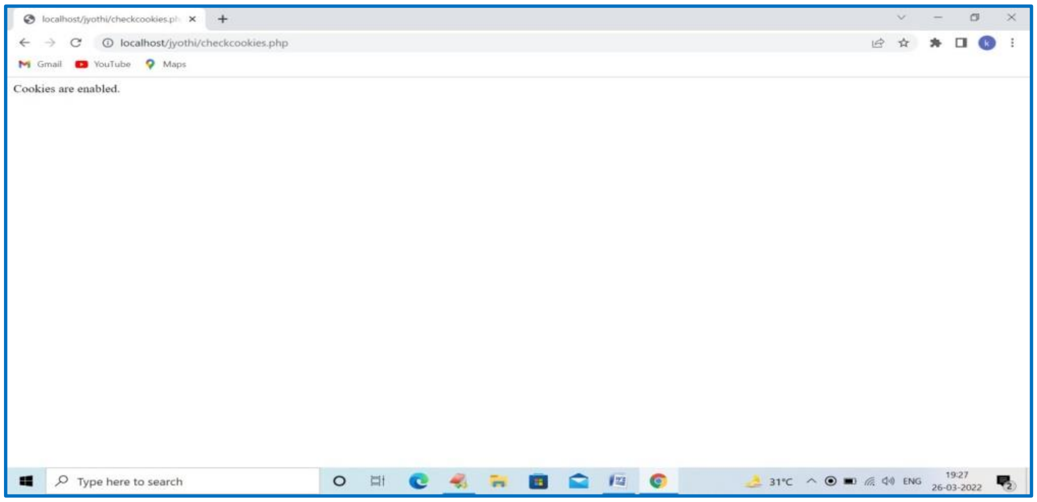


```
Cookie 'COMPUTERS' is set!
Value is: CLUSTER PAPER
```

Example2 :checkcookie.php

```
<?php
setcookie("test_cookie", "test");
?>
<html>
<body>
<?php if(count($_COOKIE) > 0)
{
 echo "Cookies are enabled.";
}
else
{
 echo "Cookies are disabled.";
}
?>
</body>
</html>
```

Output:



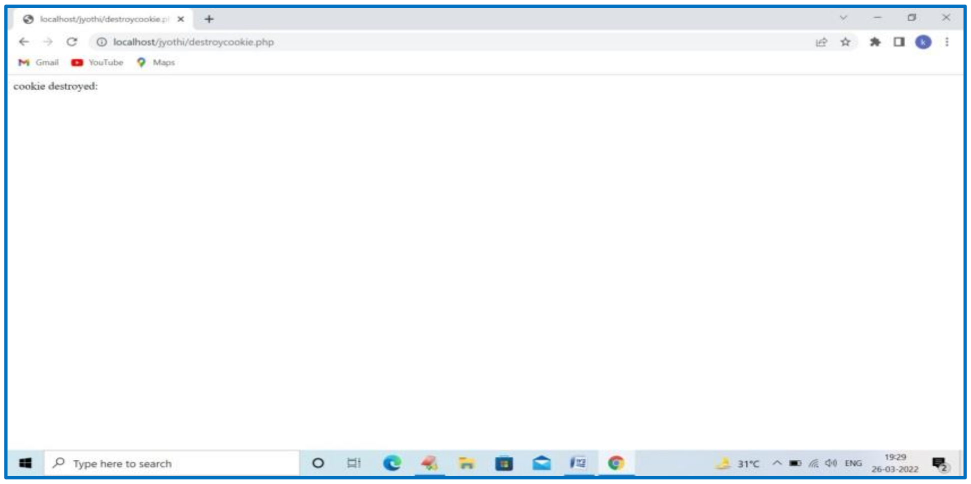
```
Cookies are enabled.
```

Q) Removing Cookies

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 the set the expiration date in the past, as shown in the example below:

EXAMPLE: destroycookie.php

```
<?php
// Deleting a cookie setcookie("username", "", time()-
3600);
?>
```


cookie destroyed:

Q) **Setting a Cookie with PHP ?**

You can set a cookie in a PHP script in two ways. First, you could use the header() function to set the Set-Cookie header.
The header() function requires a string that will then be included in the header section of the server response. Because headers are sent automatically for you, header() must be called before any output at all is sent to the browser:

header ("Set-Cookie: vegetable=artichoke; expires=Tue, 07-Mar-06 14:39:58 GMT; path=/; domain=yourdomain.com");

Although not difficult, this method of setting a cookie would require you to build a function to construct the header string. Although formatting the date as in this example and URL-encoding the name/value pair would not be a particularly arduous task, it would be a repetitive one because PHP provides a function that does just thatsetcookie().

The setcookie() function does what its name suggestsit outputs a Set-Cookie header. For this reason, it should be called before any other content is sent to the browser. The function accepts the cookie name, cookie value, expiration date in UNIX epoch format, path, domain, and integer that should be set to 1 if the cookie is only to be sent over a secure connection. All arguments to this function are optional apart from the first (cookie name) parameter.

Listing 12.1 uses setcookie() to set a cookie.

```
<?php
setcookie("vegetable", "artichoke", time()+3600, "/",
".yourdomain.com", 0);
if (isset($_COOKIE["vegetable"]))
{ echo "<p>Hello again, you have chosen:
".$_COOKIE["vegetable"].".</p>";
}
```

```
else { echo "<p>Hello you. This may be your first visit.</p>";
}
?>
```

Even though we set the cookie (line 2) when the script is run for the first time, the `$_COOKIE["vegetable"]` variable will not be created at this point. Because a cookie is read only when the browser sends it to the server, we won't be able to read it until the user revisits a page within this domain.

We set the cookie name to `"vegetable"` on line 2 and the cookie value to `"artichoke"`. We use the `time()` function to get the current time stamp and add 3600 to it (there are 3,600 seconds in an hour). This total represents our expiration date. We define a path of `"/"`, which means that a cookie should be sent for any page within our server environment. We set the domain argument to `".yourdomain.com"` (you should make the change relevant to your own domain or use `localhost`), which means that a cookie will be sent to any server in that group. Finally, we pass `0` to `setcookie()`, signaling that cookies can be sent in an insecure environment.

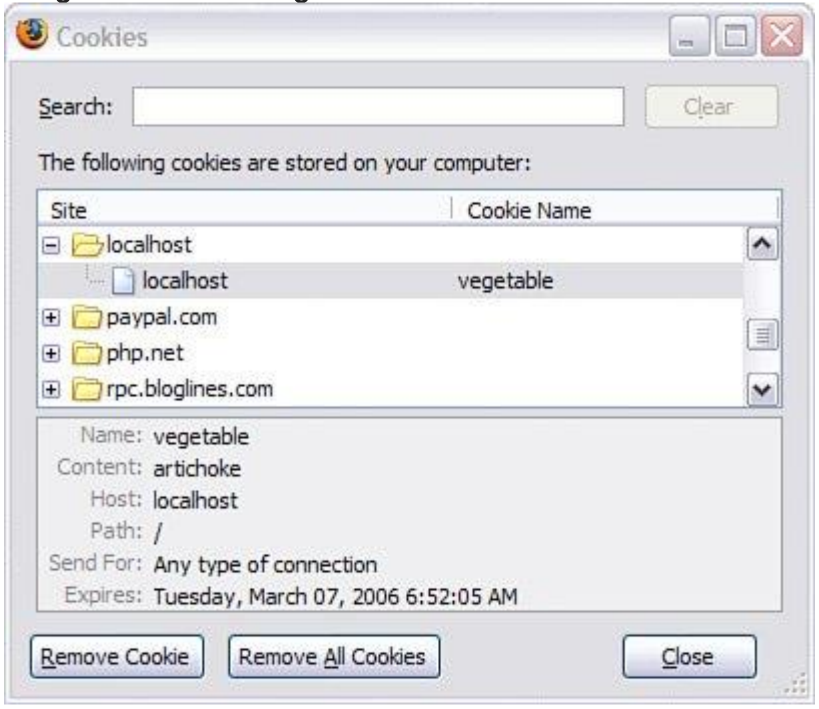
Passing `setcookie()` an empty string (`""`) for string arguments or 0 for integer fields causes these arguments to be skipped.

By the Way

With using a dynamically created expiration time in a cookie, as in Listing 12.1, note the expiration time is created by adding a certain number of seconds to the current system time of the machine running Apache and PHP. If this system clock is not accurate, it is possible that it may send in the cookie an expiration time that has already passed.

You can view your cookies in most modern web browsers. Figure 12.1 shows the cookie information stored for Listing 12.1. The cookie name, content, and expiration date appear as expected; the domain name will differ when you run this script on your own domain.

Figure 12.1. Viewing a stored cookie in a web browser.



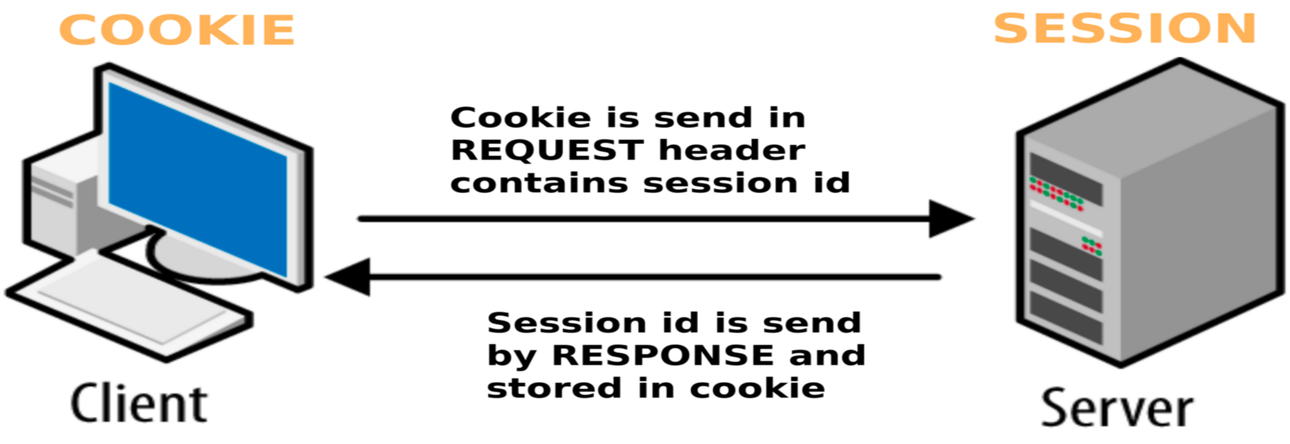
Name: vegetable
Content: artichoke
Host: localhost
Path: /
Send For: Any type of connection
Expires: Tuesday, March 07, 2006 6:52:05 AM

For more information on using cookies, and the `setcookie()` function in particular, see the PHP Manual entry at http://www.php.net/setcookie.

Q) 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.


COOKIE
SESSION
Cookie is send in REQUEST header contains session id
Session id is send by RESPONSE and stored in cookie
Client
Server

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

Start a PHP Session

- **A session is started with the session_start() function.**
- **Session variables are set with the PHP global variable: $_SESSION.**

Now, let's create a new page called "session1.php". In this page, we start a new PHP session and set some session variables:

```
 <?php session_start();
 ?>
 <html>
 <body>
 <?php
$ses=$_SESSION["user"] = "COMPUTER CLUSTER";
                              // user creates with specific name
```

```
  echo "Session information are set successfully.<br/>";
  ?>
  <a href="session2.php">Visit next page</a>
  </body>
  </html>
```

PHP Session Variable Values

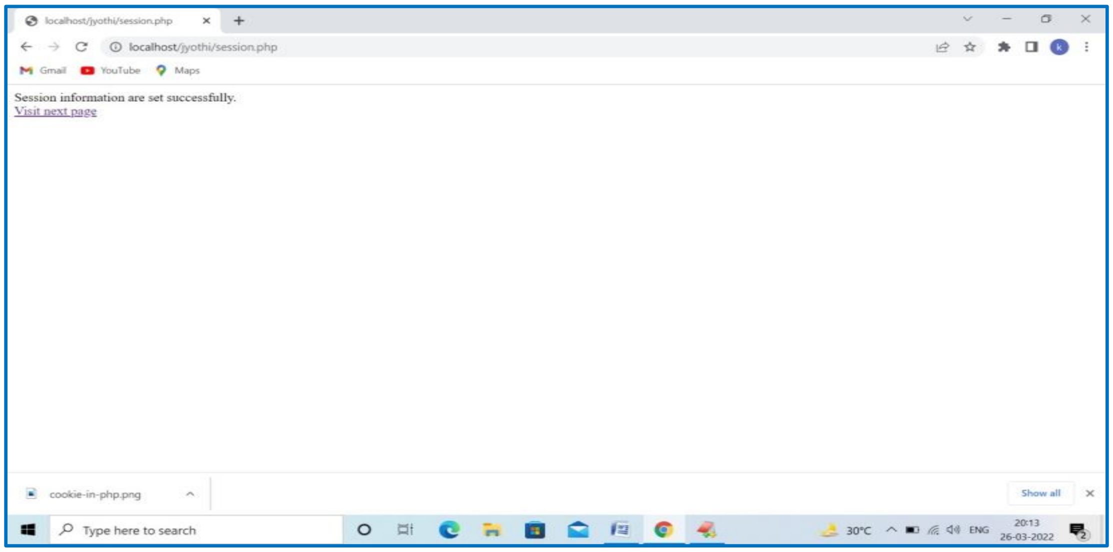
Next, we create another page called "session2.php". From this page, we will access the session information we set on the first page ("session.php").

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

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

Session2.php

```
<?php
  session_start();
  ?>
  <html>
  <body>
  <?php
  echo "User is: ".$_SESSION["user"];

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



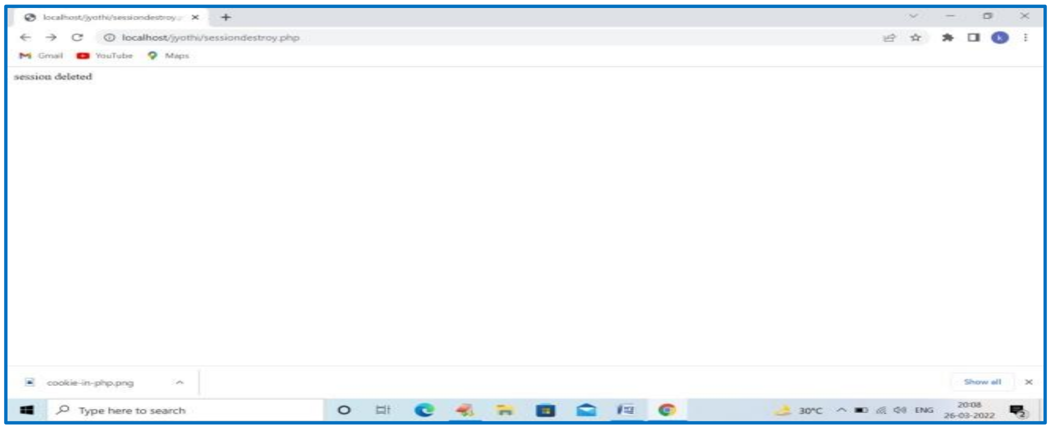
localhost/jyothi/session.php

```
Session information are set successfully.
Visit next page
```

Destroying Sessions and Unsetting Variables

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

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



localhost/jyothi/sessiondestroy.php

```
session deleted
```

You can use session_destroy() to end a session, erasing all session variables. The session_destroy() function requires no arguments. You should have an established session for this function to work as expected. The following code fragment resumes a session and abruptly destroys it:

```
session_start(); session_destroy();
```

When you move on to other pages that work with a session, the session you have destroyed will not be available to them, forcing them to initiate new sessions of their own. Any registered variables will be lost.

The session_destroy() function does not instantly destroy registered variables, however. They remain accessible to the script in which session_destroy() is called (until it is reloaded). The following

code fragment resumes or initiates a session and registers a variable called test, which we set to 5. Destroying the session does not destroy the registered variable.

session_start(); $_SESSION["test"] = 5; session_destroy(); echo $_SESSION["test"]; // prints 5

To remove all registered variables from a session, you simply unset the variable:

session_start(); $_SESSION["test"] = 5; session_destroy(); unset($_SESSION["test"]); echo $_SESSION["test"]; // prints nothing.

Using Sessions in an Environment with Registered Users

Working with Registered Users

Suppose that you've created an online community, or a portal, or some other type of application that users can "join." The process usually involves a registration form, where the user creates a username and password and completes an identification profile. From that point forward, each time a registered user logs in to the system, you can grab the user's identification information and store it in the user's session.

The items you decide to store in the user's session should be those items you can imagine using quite a bitand that would be inefficient to continually extract from the database. For example, suppose that you have created a portal in which users are assigned a certain level, such as administrator, registered user, anonymous guest, and so forth. Within your display modules, you would always want to check to verify that the user accessing the module has the proper permissions to do so. Thus, "user level" would be an example of a value stored in the user's session, so that the authentication script used in the display of the requested module only has to check a session variablethere would be no need to connect to, select, and query the database.

Working with User Preferences

If you were feeling adventurous in the design phase of a user-based application, you might have built a system in which registered users could set specific preferences that would affect the way they viewed your site. For example, you may allow your users to select from a predetermined color scheme, font type and size, and so forth. Or, you may allow users to turn "off" (or "on") the visibility of certain content groupings.

Each of those functional elements could be stored in a session. When the user logs in, the application would load all relevant values into the user's session and would react accordingly for each subsequently requested page. Should the user decide to change her preferences, she could do so while logged inyou could even prepopulate a "preferences" form based on the items stored in the session rather than going back to the database to retrieve them. If the user changes any preferences while she is logged in, simply replace the value stored in the $_SESSION superglobal with the new selectionno need to force the user to log out and then log back in again.

Passing Session IDs in the Query String

the session ID between script requests. On its own, this method is not the most reliable way of saving state because you cannot be sure that the browser will accept cookies. You can build in a failsafe, however, by passing the session ID from script to script embedded in a query string. PHP makes a name/value pair available in a constant named SID if a cookie value for a session ID cannot be found. You can add this string to any HTML links in session-enabled pages:

```
<a href="page2.html?<?php echo SID; ?>">Another page</a>
```

It will reach the browser as

```
<a
href="page2.html?PHPSESSID=08ecedf79fe34561fa82591401a01da1"
>Another page</a>
```

The session ID passed in this way will automatically be recognized in the target page when session_start() is called, and you will have access to session variables in the usual way.

Difference between cookie and session

Cookie	Session
Cookies are client-side files on a local computer that hold user information.	Sessions are server-side files that contain user data.
Cookies end on the lifetime set by the user.	When the user quits the browser or logs out of the programmed, the session is over.
It can only store a certain amount of info.	It can hold an indefinite quantity of data.
The browser's cookies have a maximum capacity of 4 KB.	We can keep as much data as we like within a session, however there is a maximum memory restriction of 128 MB that a script may consume at one time.
Because cookies are kept on the local computer, we don't need to run a function to start them.	To begin the session, we must use the session start() method.

Unit-4:: Working with Files and Directories, Images

Q) Explain about include files with include()

Including a PHP File into Another PHP File

PHP allows you to include file so that a page content can be reused many times. There are two ways to include file in PHP.

- include
- require

Advantage

Code Reusability: By the help of include and require construct, we can reuse HTML code or PHP script in many PHP scripts.

PHP include example

PHP include is used to include file on the basis of given path. Let's see a simple PHP include example.

File: footer.php

```
<?php
Echo “copy right by N Murali Krishna Mtech ”;
?>
```

File: include1.html

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

require():-

The include() and require() statement allow you to include the code contained in a PHP file within another PHP file. Including a file produces the same result as copying the script from the file specified and pasted into the location where it is called.

You can save a lot of time and work through including files — Just store a block of code in a separate file and include it wherever you want by include() and require() statement instead of typing the entire block of code multiple times.

```
<?php require "footer.php"; ?>
<html>
<head>
   <title>mypage</title>
</head>
<body>
   <h1>Welcome to Our Website!</h1>
   <p>Here you will find lots of useful information.</p>
   <?php include "footer.php"; ?>
   </body>
   </html>
```

Difference Between Include and Require Statements

if we can include files using the include() statement then why we

need require(). Typically the require() statement operates like include().
The only difference is — the include() statement will only generate a PHP warning but allow script execution to continue if the file to be included can't be found, whereas the require()statement will generate a fatal error and stops the script execution.

1. Working with Files in PHP

PHP is a server side programming language, it allows you to work with files and directories stored on the web server. We can create, access, and manipulate files on your web server using the PHP file system functions.

- **fopen()**
- **fread()**
- **fwrite()**
- **rename()**
- **fclose()**

Opening a File with PHP fopen() Function

To work with a file you first need to open the file. The PHP fopen() function is used to open a file.

The basic syntax:-

```
fopen(filename, mode)
```

The first parameter passed to fopen() specifies the name of the file you want to open, and the second parameter specifies in which mode the file should be opened.

For example:

```
<?php
$handle = fopen("data.txt", "r");
if($handle)
{
   echo "File opened successfully.";
   // Closing the file handle
   fclose($handle);
}
?>
```

The file may be opened in one of the following modes:

Modes	What it does
r	Open the file for reading only.
r+	Open the file for reading and writing.
w	Open the file for writing only and clears the contents of file. If the file does not exist, PHP will attempt to create it.

Modes	What it does
w+	Open the file for reading and writing and clears the contents of file. If the file does not exist, PHP will attempt to create it.
a	Append. Opens the file for writing only. Preserves file content by writing to the end of the file. If the file does not exist, PHP will attempt to create it.
a+	Read/Append. Opens the file for reading and writing. Preserves file content by writing to the end of the file. If the file does not exist, PHP will attempt to create it.

Reading from Files with PHP fread() Function

Now that you have understood how to open and close files. In the following section you will learn how to read data from a file. PHP has several functions for reading data from a file. You can read from just one character to the entire file with a single operation. Reading Fixed Number of Characters

The fread() function can be used to read a specified number of characters from a file.

The basic syntax:-

fread(file handle, length in bytes)

```
<?php
$file = "data.txt";
 // Check the existence of file
if(file_exists($file))
{
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the file.");
    // Read fixed number of bytes from the file
    $content = fread($handle, "20");

    // Closing the file handle
    fclose($handle);
   // Display the file content
    echo $content;
}
else
{
    echo "ERROR: File does not exist.";
}
?>
```

Writing the Files Using PHP fwrite() Function

Similarly, you can write data to a file or append to an existing file using the PHP fwrite()function.

The basic syntax:-

```
fwrite(file handle, string)
```

The fwrite() function takes two parameter — A file handle and the string of data that is to be written, as demonstrated in the following example:

```
 <?php
$file = "note.txt";

// String of data to be written
$data = "hi i am a lecturer in shdc";

// Open the file for writing
$handle = fopen($file, "w");

// Write data to the file
fwrite($handle, $data);

// Closing the file handle
fclose($handle);

echo "Data written to the file successfully.";
?>
```

Renaming Files with PHP rename() Function:

You can rename a file or directory using the PHP's rename() function, like this:

```
<?php
$file = "note.txt";

// Check the existence of file
if(file_exists($file))
{
    // Attempt to rename the file
    if(rename($file, "newfile.txt"))
    {
        echo "File renamed successfully.";
    }
  else
    {
        echo "ERROR: File cannot be renamed.";
    }
}
else
```

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

Closing a File with PHP fclose() Function

Once you've finished working with a file, it needs to be closed. The fclose() function is used to close the file, as shown in the following example:

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
   // Open the file for reading
   $handle = fopen($file, "r");
   /* Some code to be executed */
   // Closing the file handle
   fclose($handle);
}
else
{
   echo "ERROR: File does not exist.";
}
?>
```

PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax

```
unlink ( filename) ;
```

Example

```
<?php
unlink('data.txt');
echo "File deleted successfully";
?>
```

PHP Append to File

You can append data into file by using a or a+ mode in fopen() function. Let's see a simple example that appends data into data.txt file.

Let's see the data of file first.

data.txt

```
welcome to php file write
```

PHP Append to File - fwrite()

The PHP fwrite() function is used to write and append data into file.

Example

```
<?php
$fp = fopen('data.txt', 'a');//opens file in append mode
```

```
fwrite($fp, ' this is additional text ');
fwrite($fp, 'appending data');
fclose($fp);
  echo "File appended successfully";
?>
```

Q)Create a new directory in php (or) explain how to work with directories.

Working with Directories in PHP

PHP also allows you to work with directories on the file system, for example, you can open a directory and read its contents, create or delete a directory, list all files in the directory, and so on.

Creating a New Directory

You can create a new and empty directory by calling the PHP mkdir() function with the path and name of the directory to be created, as shown in the example below:

```
<?php
// The directory path
$dir = "testdir";
// Check the existence of directory
if(!file_exists($dir))
{
    // Attempt to create directory
    if(mkdir($dir))
    {
        echo "Directory created successfully.";
    }
    else
    {
        echo "ERROR: Directory could not be created.";
    }
}
else
 {
    echo "ERROR: Directory already exists.";
 }
?>
```

Copying Files from One Location to Another

We can copy a file from one location to another by calling PHP copy() function with the file's source and destination paths as arguments. If the destination file already exists it'll be overwritten. Here's an example which creates a copy of "example.txt" file inside backup folder.

```
<?php
// Source file path
$file = "example.txt";
 // Destination file path
```

```
$newfile = "backup/example.txt";
 // Check the existence of file
if(file_exists($file))
{
    // Attempt to copy file
    if(copy($file, $newfile))
    {
        echo "File copied successfully.";
    }
  else
   {
        echo "ERROR: File could not be copied.";
    }
}
else
{
    echo "ERROR: File does not exist.";
}
?>
```

To make this example work, the target directory which is backup and the source file i.e. "example.txt" has to exist already; otherwise PHP will generate an error.

Q) Explain about running commands.

System():-

system — Execute an external program and display the output.

Description

string system (string $command [, int &$return_var])

Parameters :-

command

The command that will be executed.

return_var

If the return_var argument is present, then the return status of the executed command will be written to this variable.

Return Values

Returns the last line of the command output on success,
and FALSE on failure.

Examples

```
<?php
echo '<pre>';
// Outputs all the result of shellcommand "ls", and returns
// the last output line into $last_line. Stores the return value
// of the shell command in $retval.
$last_line = system('ls', $retval);
// Printing additional info
echo '</pre>’;
<hr />Last line of the output: ' . $last_line . '
```

<hr />Return value: ' . $retval;
?>

Exec():-

exec — Execute an external program.

Description:-

string exec (string $command [, array &$output [, int &$return_var]])

exec() executes the given command.

```
<?php
// outputs the username that owns the running php/httpd process
// (on a system with the "whoami" executable in the path)
echo exec('whoami');
?>
```

Passthru():-

passthru — Execute an external program and display raw output.

Description

void passthru (string $command [, int &$return_var])

example:-

```
<?php
$filename = "backup-" . date("d-m-Y");
$cmd = "mysqldump -u root dudh_society >c:/Backup/$filename.sql";
passthru( $cmd );
if(passthru($cmd) == true)
{
   echo "Backup Succesfully";
}
else
{
   echo "Backup failed";
}
?>
```

Q) Working with images in php

Before we begin drawing on our image, there are two functions that we should consider, for added variety.

1. Line color can be modified using the imagecolorallocate() function, which we learned about before. It should be stored in a variable to be used later.
2. Line thickness can be modified using the imagesetthickness() function, which requires two parameters: imagesetthickness(image, thickness)

The imageline() function itself requires 6 parameters.

The syntax is:

imageline(image, x1, y1, x2, y2, color)

- image = Refers to the Image Resource That the Line Will Be Applied to
- x1 = x-coordinate For First Point
- y1 = y-coordinate For First Point
- x2 = x-coordinate For Second Point
- y2 = y-coordinate For Second Point

- color = Refers to the Line Color Identifier Created With imagecolorallocate()

create a 200x200 square (and a bit more)

```
<?php
create_image();
echo "<img src=image.png?".date("U").">";
function  create_image()
{
   $im = @imagecreate(200, 200) or die("Cannot Initialize new GD
image stream");
   $background_color = imagecolorallocate($im, 255, 255, 0);  //
yellow
   imagepng($im,"image.png");
   imagedestroy($im); } ?>
```

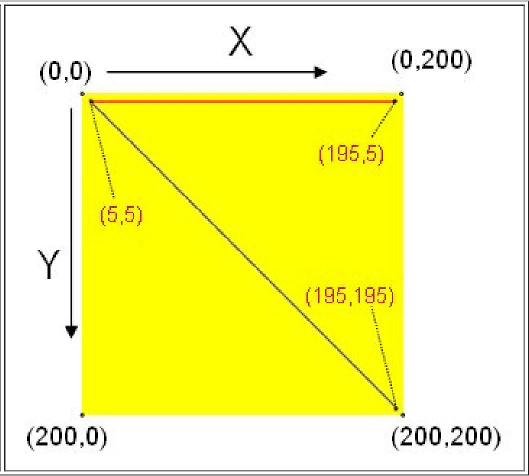
Draw line:-

```
<?php
create_image();
print "<img src=image.png?".date("U").">";
function  create_image()
{
 $im = @imagecreate(200, 200) or die("Cannot Initialize new GD
image stream");
 $background_color = imagecolorallocate($im, 255, 255, 0);   //
yellow
 $red = imagecolorallocate($im, 255, 0, 0);                 // red
 $blue = imagecolorallocate($im, 0, 0, 255);                // blue
 imageline ($im,   5,  5, 195, 5, $red);
 imageline ($im,   5,  5, 195, 195, $blue);
 imagepng($im,"image.png");
 imagedestroy($im);
}
?>
```

imageline (\$im, X_1, Y_1, X_2, Y_2, \$color);

where X_1, Y_1, X_2 and Y_2 are the positions within the image.

For example, the red line starts at
X_1=5 and Y_1=5
and ends at
X_2=195 and Y_2=5

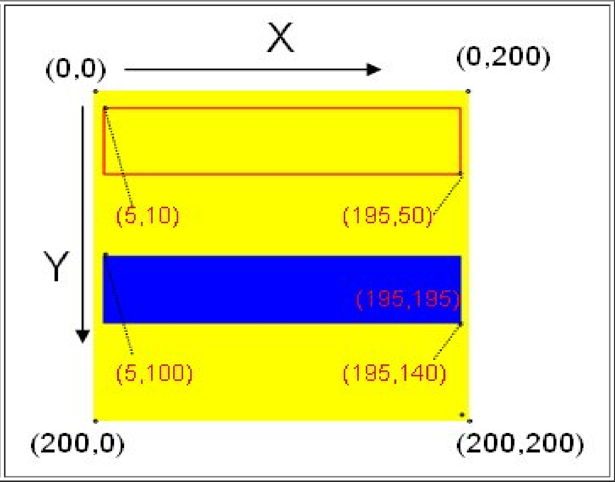


Draw rectangle:-

```
<?php
create_image();
print "<img src=image.png?".date("U").">";

function  create_image()
{
$im = @imagecreate(200, 200) or die("Cannot Initialize new GD image
stream");
 $background_color = imagecolorallocate($im, 255, 255, 0);   //
yellow
 $red = imagecolorallocate($im, 255, 0, 0);                  // red
 $blue = imagecolorallocate($im, 0, 0, 255);                 // blue
 imagerectangle ($im,   5,  10, 195, 50, $red);
 imagefilledrectangle ($im,   5,  100, 195, 140, $blue);
 imagepng($im,"image.png");
 imagedestroy($im);
}
?>
```

imagerectangle ($im, X_1, Y_1, X_2, Y_2, $color);
imagefilledectangle ($im, X_1, Y_1, X_2, Y_2, $color);

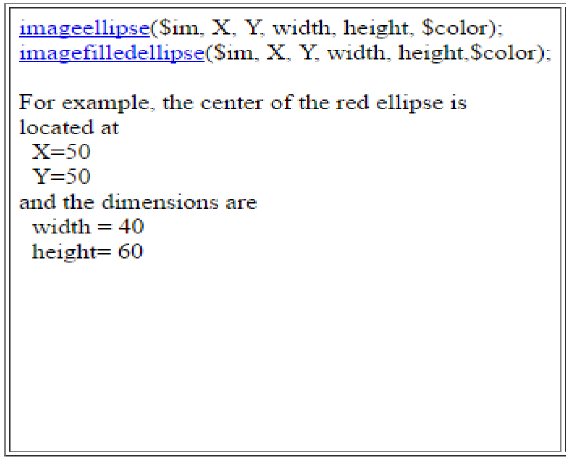
For example, the red rectangle starts at
X_1=5 and Y_1=10
and ends at
X_2=195 and Y_2=50



Draw ellipse:-

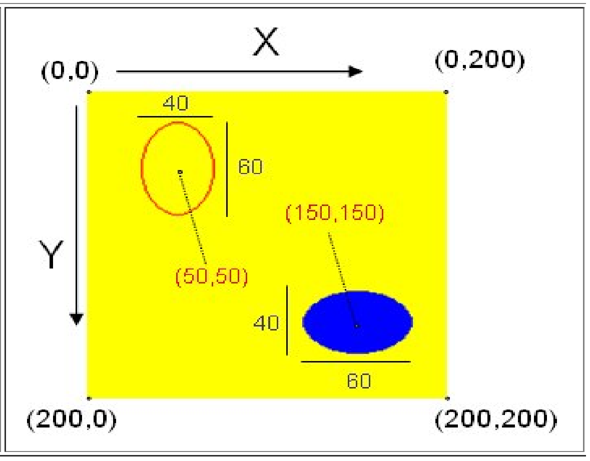
```
<?php
create_image();
print "<img src=image.png?".date("U").">";
function  create_image()
{
  $im = @imagecreate(200, 200) or die("Cannot Initialize new GD
image stream");
  $background_color = imagecolorallocate($im, 255, 255, 0);   //
yellow
```

```
  $red = imagecolorallocate($im, 255, 0, 0);                // red
  $blue = imagecolorallocate($im, 0, 0, 255);               // blue
  imageellipse($im, 50, 50, 40, 60, $red);
  imagefilledellipse($im, 150, 150, 60, 40, $blue);
  imagepng($im,"image.png");
  imagedestroy($im);
}
?>
```



imageellipse($im, X, Y, width, height, $color);
imagefilledellipse($im, X, Y, width, height,$color);

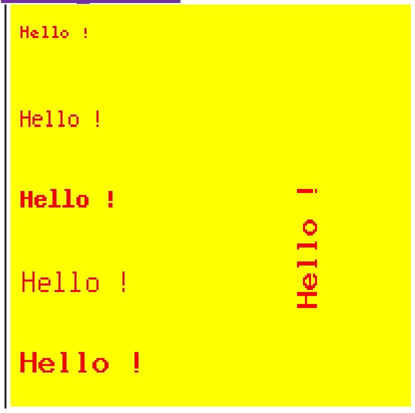
For example, the center of the red ellipse is located at
X=50
Y=50
and the dimensions are
width = 40
height= 60



Add text to the image:-

```
<?php
create_image();
print "<img src=image.png?".date("U").">";
function  create_image(){
$im = @imagecreate(200, 200)or die("Cannot Initialize new GD image
stream");
 $background_color = imagecolorallocate($im, 255, 255, 0);  // yellow
 $red = imagecolorallocate($im, 255, 0, 0);      // red
imagestring($im,
1,file:///C:/apache2triad/htdocs/0image/tutorial1.html 5,  10,
"Hello !", $red);
imagestring($im, 2, 5,  50, "Hello !", $red);
imagestring($im, 3, 5,  90, "Hello !", $red);
imagestring($im, 4, 5, 130, "Hello !", $red);
imagestring($im, 5, 5, 170, "Hello !", $red);
imagestringup($im, 5, 140, 150, "Hello !", $red);
imagepng($im,"image.png");
imagedestroy($im);
}
?>
```

output:-

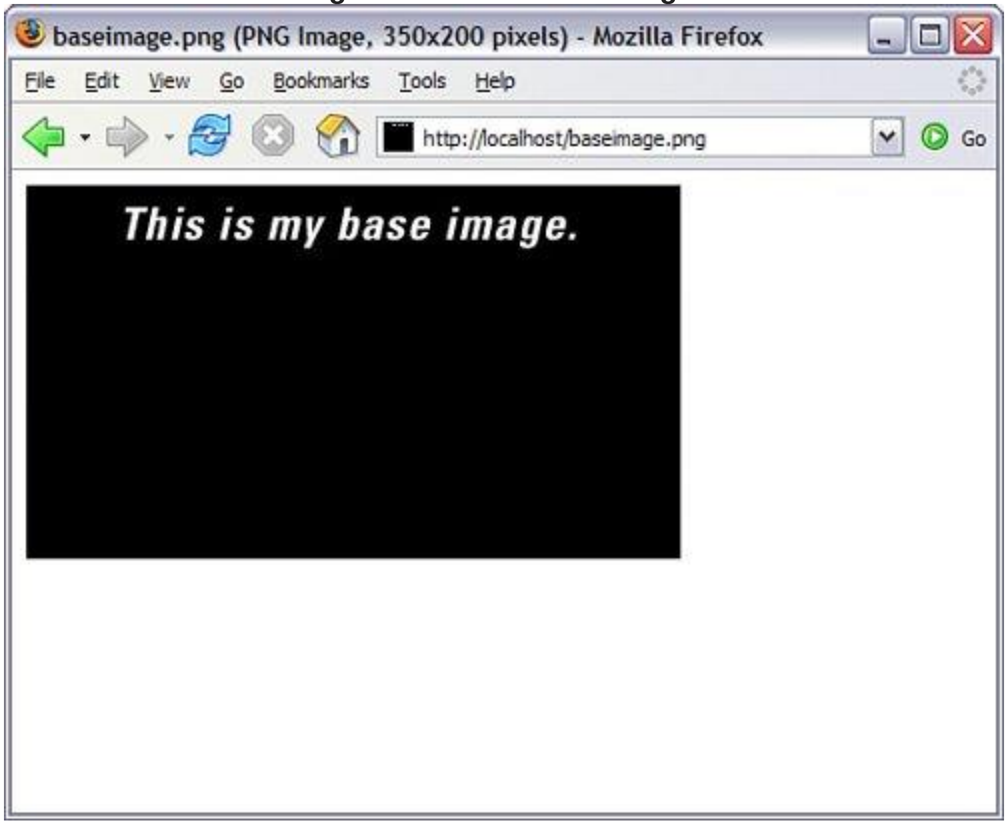


Modifying Existing Images

The process of creating images from other images follows the same essential steps as creating a new imagethe difference lies in what acts as the image canvas. Previously, you created a new canvas using the `ImageCreate()` function. When creating an image from a new image, you use the `ImageCreateFrom*()` family of functions.

You can create images from existing GIFs, JPEGs, PNGs, and plenty of other image types. The functions used to create images from these formats are called `ImageCreateFromGif()`, `ImageCreateFromJpg()`, `ImageCreateFromPng()`, and so forth. In the next example, you can see how easy it is to create a new image from an existing one. Figure 14.5 shows the base image.

Figure 14.5. The base image.


baseimage.png (PNG Image, 350x200 pixels) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/baseimage.png
Go
This is my base image.

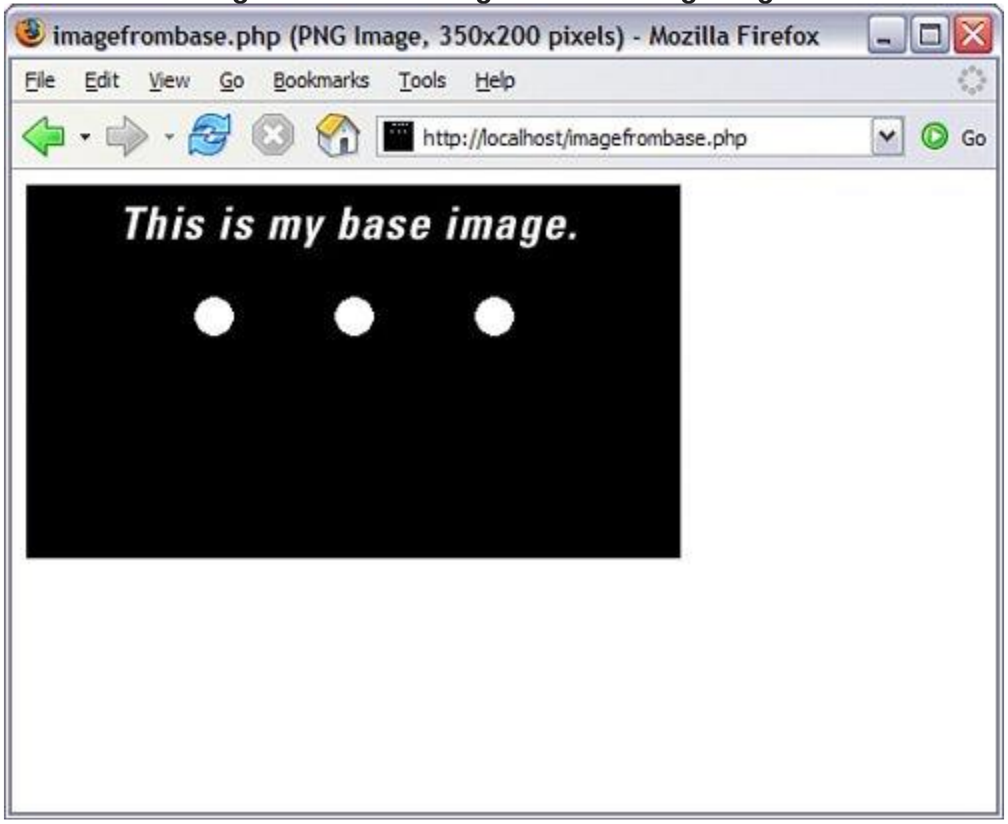
Listing 14.5 shows how to use an existing image as the canvas, which then has ellipses drawn on it.

Listing 14.5. Creating a New Image from an Existing Image

```
<?php
//use existing image as a canvas
$myImage = ImageCreateFromPng("baseimage.png");

//allocate the color white
$white = ImageColorAllocate($myImage, 255, 255, 255);

//draw on the new canvas
ImageFilledEllipse($myImage, 100, 70, 20, 20, $white);
ImageFilledEllipse($myImage, 175, 70, 20, 20, $white);
ImageFilledEllipse($myImage, 250, 70, 20, 20, $white);

//output the image to the browser
header ("Content-type: image/png");
ImagePng($myImage);

//clean up after yourself
ImageDestroy($myImage);
?>
```

Save this listing as `imagefrombase.php` and place it in the document root of your web server. When accessed, it should look something like Figure 14.6.

Figure 14.6. Drawing on an existing image.


imagefrombase.php (PNG Image, 350x200 pixels) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/imagefrombase.php
Go
This is my base image.

Getting fancy with pie charts?

You can use this same sequence of events to expand your scripts to create charts and graphs, using either static or dynamic data for the data points. Listing 14.3 draws a basic pie chart. Lines 1 through 10 will look exactly the same as the previous listings, because they just set up the canvas size and colors to be used.

Listing 14.3. A Basic Pie Chart

```
 1:  <?php
2:  //create the canvas
3:  $myImage = ImageCreate(300,300);
4: 5:  //set up some colors
6:  $white = ImageColorAllocate($myImage, 255, 255, 255);
7:  $red  = ImageColorAllocate($myImage, 255, 0, 0);
 8:  $green = ImageColorAllocate($myImage, 0, 255, 0);
9:  $blue = ImageColorAllocate($myImage, 0, 0, 255);
10:
11: //draw a pie
12: ImageFilledArc($myImage, 100,100,200,150,0,90, $red,
IMG_ARC_PIE);
13: ImageFilledArc($myImage, 100,100,200,150,90,180,
$green, IMG_ARC_PIE);
14: ImageFilledArc($myImage, 100,100,200,150,180,360,
$blue, IMG_ARC_PIE);
15: 16: //output the image to the browser
17: header ("Content-type: image/png");
 18: ImagePng($myImage);
19: 20: //clean up after yourself
21: ImageDestroy($myImage);
22: ?>
```

Okay, so the definition of the color black has been removed from this example, but it's mostly the same. Because we have removed the definition of black, the first defined color is white. Therefore, the color of the canvas will be white.

In lines 12&14, we use the `ImageFilledArc()` function, which has several attributes:

- ✓ The image identifier
- ✓ The partial ellipse centered at x
- ✓ The partial ellipse centered at y
- ✓ The partial ellipse width
- ✓ The partial ellipse height
- ✓ The partial ellipse start point
- ✓ The partial ellipse end point
- ✓ Color
- ✓ Style

Look at line 14 from Listing 14.3:

```
14: ImageFilledArc($myImage, 100,100,200,150,180,360, $blue,
IMG_ARC_PIE);
```

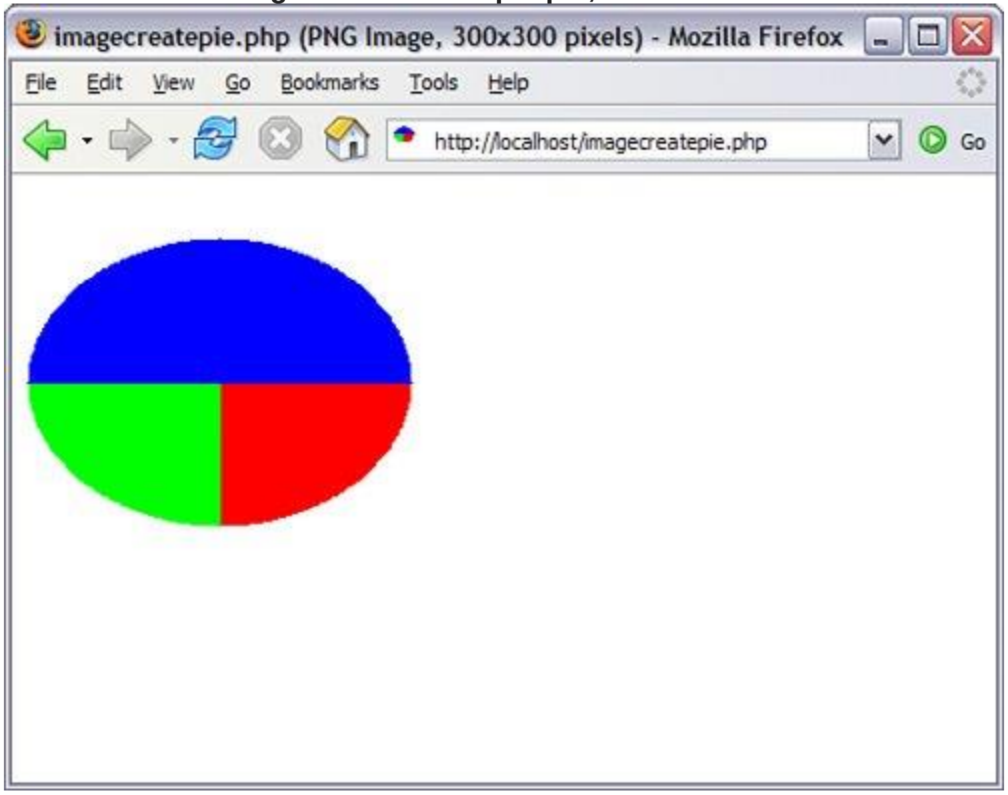
The arc should be filled with the defined color `$blue` and should use the `IMG_ARC_PIE` style. The `IMG_ARC_PIE` style is one of several built-in styles used in the display; this one says to create a rounded edge.

By the Way

You can learn about all the various styles in the PHP manual, at http://www.php.net/image.

Save this listing as `imagecreatepie.php` and place it in the document root of your web server. When accessed, it should look something like Figure 14.3, but in color.

Figure 14.3. A simple pie, with slices.


imagecreatepie.php (PNG Image, 300x300 pixels) - Mozilla Firefox
File
Edit
View
Go
Bookmarks
Tools
Help
http://localhost/imagecreatepie.php
Go

You can extend the code in Listing 14.3 and give your pie a 3D appearance. To do so, define three more colors for the edge. These colors can be either lighter or darker than the base colors, as long as they provide some contrast. The following examples define lighter colors:

```
$lt_red = ImageColorAllocate($myImage, 255, 150, 150);
$lt_green = ImageColorAllocate($myImage, 150, 255, 150);
$lt_blue = ImageColorAllocate($myImage, 150, 150, 255);
```

To create the shading effect, you use a `for` loop to add a series of small arcs at the points (100,120) to (100,101), using the lighter colors as fill colors:

```
for ($i = 120;$i > 100;$i--) {     ImageFilledArc ($myImage,
100,$i,200,150,0,90, $lt_red, IMG_ARC_PIE);     ImageFilledArc
($myImage, 100,$i,200,150,90,180, $lt_green, IMG_ARC_PIE);
ImageFilledArc ($myImage, 100,$i,200,150,180,360, $lt_blue,
IMG_ARC_PIE); }
```

Image Creation from User Input ?

to creating images from other images and drawing images on your own, you can also create images based on user input. There's no fundamental difference in how the scripts are created except for the fact that you'll be gathering values from a form instead of hard-coding them into your script.

In Listing 14.7, we create an all-in-one form and script, which asks for user input for a variety of attributes ranging from image size to text and background colors, as well as a message string. You'll be introduced to the `imagestring()` function, which is used to "write" a string onto an image.

```
if (!$_POST) {
    //show form
    echo "
<html>
    <head>
    <title>Image Creation Form</title>
    </head>
    <body>
<h1>Create an Image</h1>
   <form method=\"POST\" action=\"".$_SERVER["PHP_SELF"]."\">
<p><strong>Image Size:</strong><br/>
   W: <input type=\"text\" name=\"w\" size=\"5\" maxlength=\"5\" />
   H: <input type=\"text\" name=\"h\" size=\"5\" maxlength=\"5\" /></p>
   <p><strong>Background Color:</strong><br />
   R: <input type=\"text\" name=\"b_r\" size=\"3\" maxlength=\"3\" />
   G: <input type=\"text\" name=\"b_g\" size=\"3\" maxlength=\"3\" />
   B: <input type=\"text\" name=\"b_b\" size=\"3\" maxlength=\"3\" /></p>
<p><strong>Text Color:</strong><br />
   R: <input type=\"text\" name=\"t_r\" size=\"3\" maxlength=\"3\" />
  G: <input type=\"text\" name=\"t_g\" size=\"3\" maxlength=\"3\" />
  B: <input type=\"text\" name=\"t_b\" size=\"3\" maxlength=\"3\" /></p>
<p><strong>Text String:</strong><br />
  <input type=\"text\" name=\"string\" size=35 /></p>
<p><strong>Font Size:</strong><br />
  <select name=\"font_size\">
  <option value=\"1\">1</option>
  <option value=\"2\">2</option>
  <option value=\"3\">3</option>
  <option value=\"4\">4</option>
  <option value=\"5\">5</option>
  </select></p>
<p><strong>Text Starting Position:</strong><br />
X: <input type=\"text\" name=\"x\" size=\"3\" maxlength=\"3\" />
  Y: <input type=\"text\" name=\"y\" size=\"3\" maxlength=\"3\" /></p>
<p><input type=\"submit\" name=\"submit\" value=\"create image\" /></p>
  </form>
  </body>
</html>";
} else {
  //create image
//create the canvas
  $myImage = ImageCreate($_POST["w"], $_POST["h"]);

  //set up some colors
  $background = ImageColorAllocate ($myImage, $_POST["b_r"],
      $_POST["b_g"],
$_POST["b_b"]); 47:   $text = ImageColorAllocate
($myImage, $_POST["t_r"], 48:       $_POST["t_g"],
$_POST["t_b"]); 49: 50:   // write the string at the top
left 51:   ImageString($myImage, $_POST["font_size"],
$_POST["x"], 52:       $_POST["y"], $_POST["string"],
$text); 53: 54:   //output the image to the browser 55:
header ("Content-type: image/png"); 56:
ImagePNG($myImage); 57: 58:   //clean up after yourself
59:   ImageDestroy($myImage); 60: } 61: ?>
```

Let's get into the script, where lines 238 represent the user input form, and the remaining lines handle the image created per user specifications.

In this basic form, you see that several fields are used to obtain image specifications. On lines 1213 there are fields to define the width and the height of the image we want to draw. Next, we set up fields to obtain the RGB values for a background color (lines 1517) and a text color (lines 1921).

By the Way

You could create drop-down list boxes containing values 0 through 255, for the red, green, and blue values. This would ensure that the user input was within the required range.

Line 23 contains a form field for the input string. This string will be drawn onto the background of the image in the text color specified. Lines 2531 represent a drop-down list for the selection of the font size. There are five sizes, 1 through 5, for the default fixed-width font.

By the Way

You can specify fonts using the `imageloadfont()` and `imagettftext()` functions. Learn more at http://www.php.net/image.

Finally, lines 33 and 34 allow you to define the text starting position. The upper-left corner of the image area would be X position 0, Y position 0; 10 increments downward would be Y position 10, 10 increments to the right would be X position 10, and so forth.

If we stopped the script here and closed up the `if...else` statement and PHP block, we would see a form like Figure 14.8 when loaded in our web browser.

Figure 14.8. User input form for image creation.

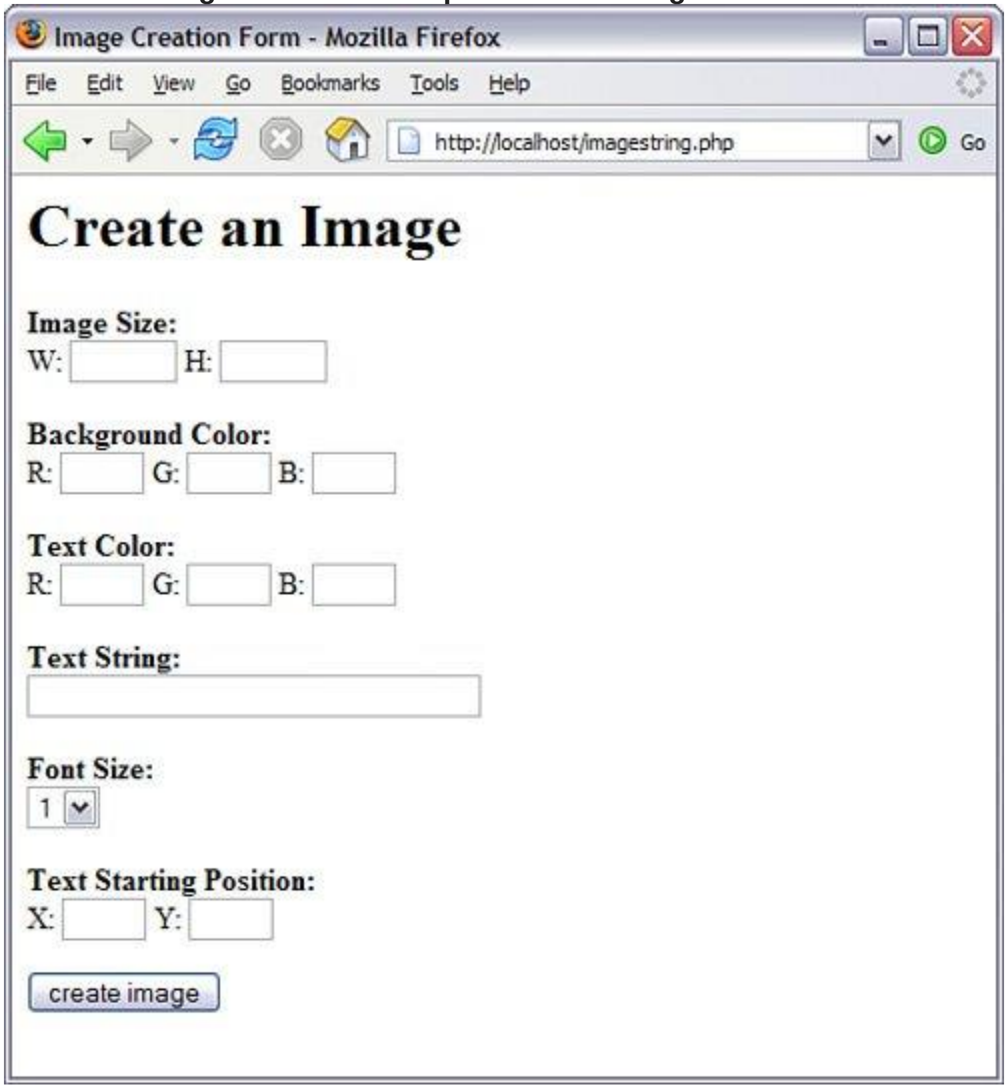


Image Creation Form - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/imagestring.php
Go

Create an Image

Image Size:
W: H:

Background Color:
R: G: B:

Text Color:
R: G: B:

Text String:

Font Size:
1

Text Starting Position:
X: Y:

create image

In only 18 more lines, we can finish this script and generate images with text strings, so take a look at the remainder of Listing 14.7.

The majority of lines 3961 you've already seen before, only this time we use extracted elements from the `$_POST` superglobal to take the place of hard-coded values. In line 42 we use the width and height values from the form to set up the initial image. Lines 4547 define two colors, `$background` and `$text`, using the appropriate RGB values provided by the form.

By the Way

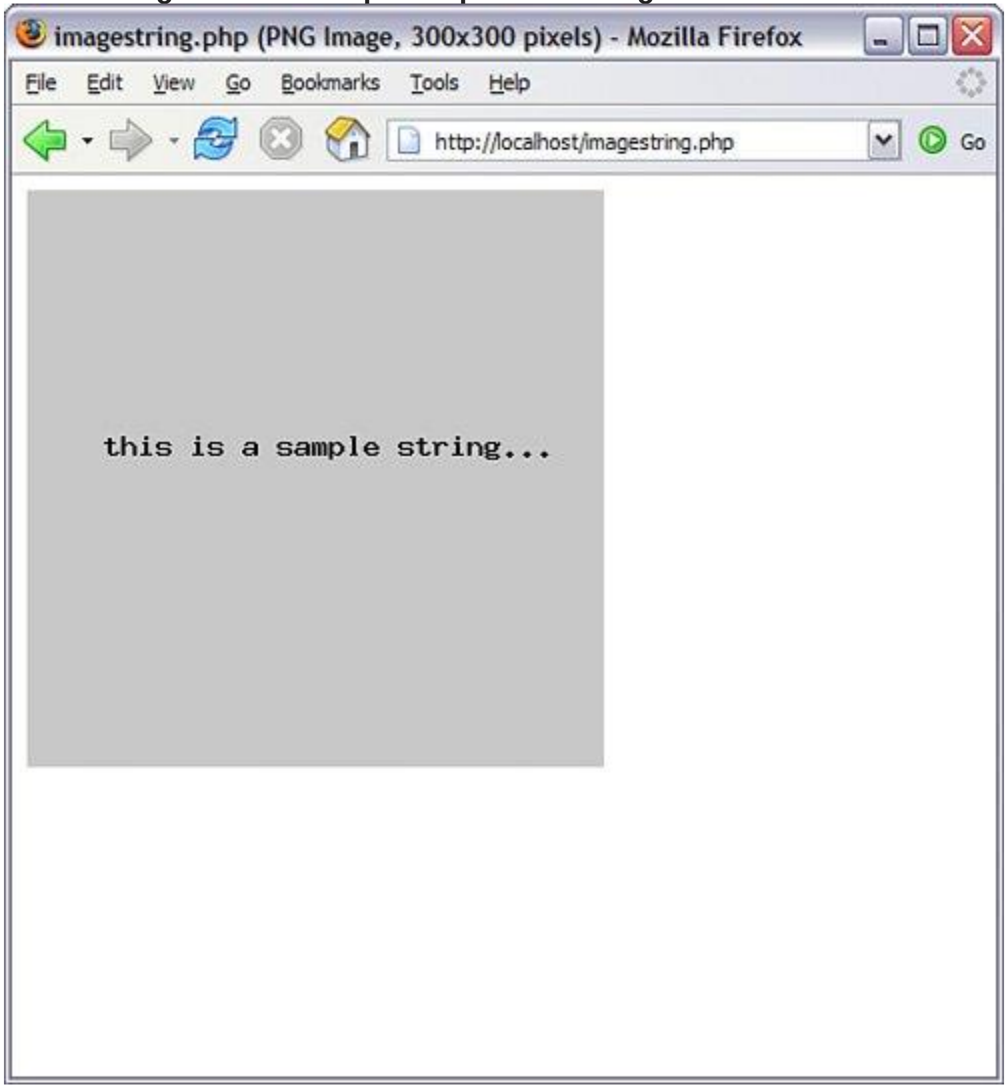
The colors weren't given actual color names in this script because we don't know what the user input would createwe could call the color `$red`, but if they defined it as 0,255,0 we'd look stupid because that's the RGB value for green! Instead, we simply name our colors after their purpose, not their appearance.

Lines 5152 represent the only new item in this script, the use of the `imagestring()` function. The six parameters for this function are the image stream (`$myImage`), the font size (`$_POST["font_size"]`), the starting X and Y positions

(`$_POST["x"]` and `$_POST["y"]`), the string to be drawn (`$_POST["string"]`), and the color in which to draw it (`$text`). Lines 5556 output the image to the browser, and line 59 destroys and cleans up the image creation process.

If we save this file as `imagecreate.php`, place it in the document root of the web server, and fill out the form, the output could look something like Figure 14.9. But quite likely your results will differ because there are many variables to play with!

Figure 14.9. Sample output from image creation form.


imagestring.php (PNG Image, 300x300 pixels) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/imagestring.php
Go

```
this is a sample string...
```

Unit-V :: Interacting with MySQL using PHP

Q)MySQL Versus MySQLi Functions.

MySQL MySQLi Difference

There are too many differences between these PHP database extensions. These differences are based on some factors like performance, library functions, features, benefits, and others.

MySQL	MySQLi
MySQL extension added in PHP version 2.0. and deprecated as of PHP 5.5.0.	MySQLi extension added in PHP 5.5 and will work on MySQL 4.1.3 or above.
Does not support prepared statements.	MySQLi supports prepared statements.
MySQL provides the procedural interface.	MySQLi provides both procedural and object-oriented interface.
MySQL extension does not support stored procedure.	MySQLi supports store procedure.
MySQL extension lags in security and other special features, comparatively.	MySQLi extension is with enhanced security and improved debugging.
Transactions are handled by SQL queries only.	MySQLi supports transactions through API.
Extension directory: ext/mysql.	Extension directory: ext/mysqli.

Q) Explain how to connect mysql with PHP.

PHP MySQL Connect
Since PHP 5.5, mysql_connect() extension is deprecated. Now it is recommended to use one of the 2 alternatives.
mysqli_connect()
PDO::__construct()
PHP mysqli_connect()
PHP mysqli_connect() function is used to connect with MySQL database. It returns resource if connection is established or null.
Syntax
resource mysqli_connect (server, username, password)

PHP mysqli_close()
PHP mysqli_close() function is used to disconnect with MySQL database. It returns true if connection is closed or false.

Syntax

```
bool mysqli_close(resource $resource_link)
```

PHP MySQL Connect Example

Example

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

Q)How to Connect to MySQL Database Server?

In PHP you can easily do this using the mysqli_connect() function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:

Syntax: MySQLi, Procedural way

```
$link = mysqli_connect("hostname", "username", "password",
"database");
```

Syntax: MySQLi, Object Oriented way

```
$mysqli = new mysqli("hostname", "username", "password",
"database");
```

Syntax: PHP Data Objects (PDO) way

```
$pdo = new PDO("mysql:host=hostname;dbname=database",
"username", "password");
```

The hostname parameter in the above syntax specify the host name (e.g. localhost), or IP address of the MySQL server, whereas the username and password parameters specifies the credentials to access MySQL server, and the database parameter, if provided will specify the default MySQL database to be used when performing queries.

```
<?php
$link = mysqli_connect("localhost", "root", "",”shdcupdates”);
// Check connection
if($link === false){
    die("ERROR: Could not connect the data base. " .
mysqli_connect_error());
}

// Print host information
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);
```

```
?>
```

Q)How to create a MySQL Database Using PHP.

Creating MySQL Database Using PHP:-

Before saving or accessing the data, we need to create a database first. The CREATE DATABASE statement is used to create a new database in MySQL.

Let's make a SQL query using the CREATE DATABASE statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to finally create our database. The following example creates a database named demo.

Example:-

```
<?php
$link = mysqli_connect("localhost", "root", "");
// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
// Attempting to create database query execution
$sql = "CREATE DATABASE shdcupdates";
if(mysqli_query($link, $sql)){
    echo "Database created successfully";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}
// Close connection
mysqli_close($link);
?>
```

Q)How to create a Mysql database table using PHP.

We can create tables inside the database that will actually hold the data. A table organizes the information into rows and columns. The SQL CREATE TABLE statement is used to create a table in database.

Let's make a SQL query using the CREATE TABLE statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to finally create our table.

Example:-

```
<?php
/* Attempt MySQL server connection. Assuming you are running
MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Attempt create table query execution
$sql = "CREATE TABLE persons(
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    email VARCHAR(70) NOT NULL UNIQUE
)";
if(mysqli_query($link, $sql)){
    echo "Table created successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}
// Close connection
mysqli_close($link);
?>
```

Q)How to insert a record into the table using PHP.

The INSERT INTO statement is used to insert new rows in a database table.Let's make a SQL query using the INSERT INTO statement with appropriate values, after that we will execute this insert query through passing it to the PHP mysqli_query() function to insert data in table. Here's an example, which insert a new row to the persons table by specifying values for the first_name, last_name and email fields.

```
<?php
/* Attempt MySQL server connection. Assuming you are running
MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES
('Peter', 'Parker', 'peterparker@mail.com')";
if(mysqli_query($link, $sql)){
    echo "Records inserted successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}
// Close connection
mysqli_close($link);
?>
```

Q)Explain how to view records from the table using PHP.

View/Selecting Data From Database Tables

The SQL SELECT statement is used to select the records from database tables. Its basic syntax is as follows:

SELECT column1_name, column2_name, columnN_name FROM table_

name;
Let's make a SQL query using the SELECT statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to retrieve the table data.
Consider our persons database table has the following records:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

The PHP code in the following example selects all the data stored in the persons table (using the asterisk character (*) in place of column name selects all the data in the table).

```
<?php
/* Attempt MySQL server connection. Assuming you are running
MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
 // Check connection
if($link === false){
   die("ERROR: Could not connect. " . mysqli_connect_error());
}
 // Attempt select query execution
$sql = "SELECT * FROM persons";
if($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>";
      // Free 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);
}

// Close connection
mysqli_close($link);
?>
```

Q)How to delete table record using PHP.

Deleting Database Table Data

We can delete records from a table using the SQL DELETE statement. It is typically used in conjugation with the WHERE clause to delete only those records that matches specific criteria or condition. The basic syntax of the DELETE statement can be given with:

DELETE FROM table_name WHERE column_name=some_value

Let's make a SQL query using the DELETE statement and WHERE clause, after that we will execute this query through passing it to the PHP mysqli_query() function to delete the tables records.

Consider the following persons table inside the demo database:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

The PHP code in the following example will delete the records of those persons from the persons table whose first_name is equal to John.

Example Program:-

```
<?php
/* Attempt MySQL server connection. Assuming you are running
MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
```

```
// Check connection
if($link === false){
   die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Attempt delete query execution
$sql = "DELETE FROM persons WHERE first_name='John'";
if(mysqli_query($link, $sql)){
   echo "Records were deleted successfully.";
} else{
   echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

q) Creating an Online Address Book

Address Book Table and Field Names

Table Name	**Field Names**
master_name	id, date_added, date_modified, f_name, l_name
address	id, master_id, date_added, date_modified, address, city, state, zipcode, type
telephone	id, master_id, date_added, date_modified, tel_number, type
fax	id, master_id, date_added, date_modified, fax_number, type
email	id, master_id, date_added, date_modified, email, type
personal_notes	id, master_id, date_added, date_modified, note

creates the `master_name` table:

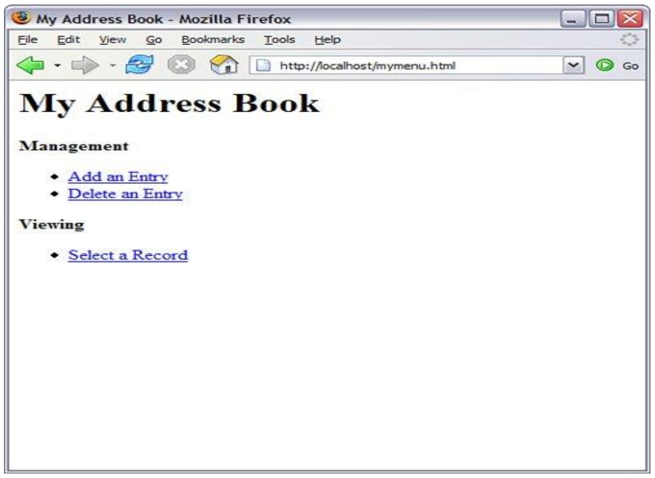
```
 mysql> CREATE TABLE master_name (       -> id INT NOT NULL
PRIMARY KEY AUTO_INCREMENT,      -> date_added DATETIME,      -
> date_modified DATETIME,      -> f_name VARCHAR (75),      ->
l_name VARCHAR (75)      -> );
```

similarly all above tables created

Address Book Menu

```
  1: <html>  2: <head>  3: <title>My Address Book</title>
4: </head>  5: <body>  6: <h1>My Address Book</h1>  7:
8: <p><strong>Management</strong></p>  9: <ul> 10: <li><a
href="addentry.php">Add an Entry</a></li> 11: <li><a
href="delentry.php">Delete an Entry</a></li> 12: </ul>
13: 14: <p><strong>Viewing</strong></p> 15: <ul> 16:
<li><a href="selentry.php">Select a Record</a></li> 17:
</ul> 18: </body> 19: </html>
```

Address book menu.


My Address Book - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/mymenu.html
Go

My Address Book

Management

- Add an Entry
- Delete an Entry

Viewing

- Select a Record

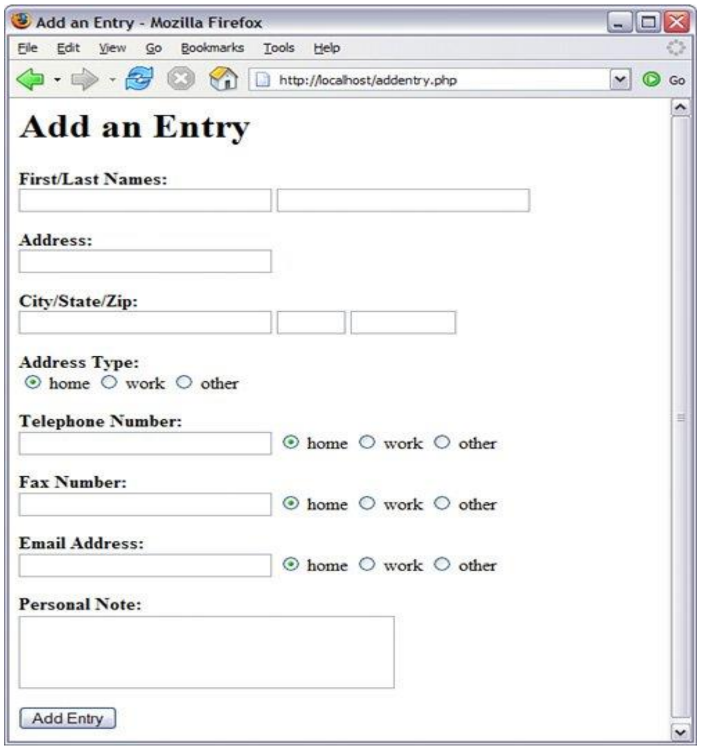
Creating the Record Addition Mechanism

Basic Record Addition Script Called `addentry.php`

```
 1:   <?php 2:   if (!$_POST) { 3:       //haven't seen
the form, so show it 4:       $display_block = " 5:
<form method=\"post\" action=\"".$_SERVER["PHP_SELF"]."\">
6:       <p><strong>First/Last Names:</strong><br/> 7:
<input type=\"text\" name=\"f_name\" size=\"30\"
maxlength=\"75\"> 8:       <input type=\"text\"
name=\"l_name\" size=\"30\" maxlength=\"75\"></p> 9: 10:
<p><strong>Address:</strong><br/> 11:      <input
type=\"text\" name=\"address\" size=\"30\"></p> 12: 13:
<p><strong>City/State/Zip:</strong><br/> 14:      <input
type=\"text\" name=\"city\" size=\"30\" maxlength=\"50\">
15:      <input type=\"text\" name=\"state\" size=\"5\"
maxlength=\"2\"> 16:      <input type=\"text\"
name=\"zipcode\" size=\"10\" maxlength=\"10\"></p> 17: 18:
<p><strong>Address Type:</strong><br/> 19:      <input
type=\"radio\" name=\"add_type\" value=\"home\" checked>
home 20:      <input type=\"radio\" name=\"add_type\"
value=\"work\"> work 21:      <input type=\"radio\"
name=\"add_type\" value=\"other\"> other</p> 22: 23:
<p><strong>Telephone Number:</strong><br/> 24:      <input
type=\"text\" name=\"tel_number\" size=\"30\"
maxlength=\"25\"> 25:      <input type=\"radio\"
name=\"tel_type\" value=\"home\" checked> home 26:
<input type=\"radio\" name=\"tel_type\" value=\"work\">
work 27:      <input type=\"radio\" name=\"tel_type\"
value=\"other\"> other</p> 28: 29:      <p><strong>Fax
Number:</strong><br/> 30:      <input type=\"text\"
name=\"fax_number\" size=\"30\" maxlength=\"25\"> 31:
<input type=\"radio\" name=\"fax_type\" value=\"home\"
checked> home 32:      <input type=\"radio\"
name=\"fax_type\" value=\"work\"> work 33:      <input
type=\"radio\" name=\"fax_type\" value=\"other\">
other</p> 34: 35:      <p><strong>Email
Address:</strong><br/> 36:      <input type=\"text\"
name=\"email\" size=\"30\" maxlength=\"150\"> 37:
<input type=\"radio\" name=\"email_type\" value=\"home\"
```

```
checked> home 38:      <input type=\"radio\"
name=\"email_type\" value=\"work\"> work 39:      <input
type=\"radio\" name=\"email_type\" value=\"other\">
other</p> 40: 41:      <p><strong>Personal
Note:</strong><br/> 42:      <textarea name=\"note\"
cols=\"35\" rows=\"3\" 43:
wrap=\"virtual\"></textarea></p> 44: 45:      <p><input
type=\"submit\" name=\"submit\" value=\"Add Entry\"></p>
46:      </form>"; 47: 48:  } else if ($_POST) { 49:
//time to add to tables, so check for required fields 59:
if (($_POST["f_name"] == "") || ($_POST["l_name"] == ""))
{ 60:          header("Location: addentry.php"); 61:
exit; 62:      } 63: 64:      //connect to database 65:
$mysqli =
mysqli_connect("localhost","joeuser","somepass","testDB");
66: 67:      //add to master_name table 68:
$add_master_sql = "INSERT INTO master_name (date_added,
date_modified, 69:                        f_name, l_name)
VALUES (now(), now(), 70:
'".$_POST["f_name"]."', '".$_POST["l_name"]."')"; 71:
$add_master_res = mysqli_query($mysqli, $add_master_sql)
72:                        or die(mysqli_error($mysqli));
73: 74:      //get master_id for use with other tables 75:
$master_id = mysqli_insert_id($mysqli); 76: 77:      if
(($_POST["address"]) || ($_POST["city"]) ||
($_POST["state"]) 78:          || ($_POST["zipcode"])) {
79:          //something relevant, so add to address table
80:          $add_address_sql = "INSERT INTO address
(master_id, date_added, 81:
date_modified, address, city, state, zipcode, 82:
type)  VALUES ('".$master_id."', now(), now(), 83:
'".$_POST["address"]."', '".$_POST["city"]."', 84:
'".$_POST["state"]."', '".$_POST["zipcode"]."', 85:
'".$_POST["add_type"]."')"; 86:          $add_address_res
= mysqli_query($mysqli, $add_address_sql) 87:
or die(mysqli_error($mysqli)); 88:      } 89: 90:      if
($_POST["tel_number"]) { 91:          //something
relevant, so add to telephone table 92:
$add_tel_sql = "INSERT INTO telephone (master_id,
date_added, 93:                        date_modified,
tel_number, type) VALUES 94:
('".$master_id."', now(), now(), 95:
'".$_POST["tel_number"]."', 96:
'".$_POST["tel_type"]."')"; 97:          $add_tel_res =
mysqli_query($mysqli, $add_tel_sql) 98:
or die(mysqli_error($mysqli)); 99:      } 100: 101:     if
($_POST["fax_number"]) { 102:         //something
relevant, so add to fax table 103:         $add_fax_sql =
"INSERT INTO fax (master_id, date_added, 104:
date_modified, fax_number, type)  VALUES 105:
('".$master_id."', now(), now(), 106:
'".$_POST["fax_number"]."', 107:
'".$_POST["fax_type"]."')"; 108:         $add_fax_res =
mysqli_query($mysqli, $add_fax_sql) 109:
or die(mysqli_error($mysqli)); 110:     } 111: 112:     if
($_POST["email"]) { 113:         //something relevant, so
add to email table 114:         $add_email_sql = "INSERT
INTO email (master_id, date_added, 115:
date_modified, email, type)  VALUES 116:
('".$master_id."', now(), now(), 117:
'".$_POST["email"]."', 118:
'".$_POST["email_type"]."')"; 119:          $add_email_res
= mysqli_query($mysqli, $add_email_sql) 120:
or die(mysqli_error($mysqli)); 121:     } 122: 123:     if
($_POST["note"]) { 124:         //something relevant, so
add to notes table 125:         $add_notes_sql = "INSERT
INTO personal_notes (master_id, date_added, 126:
date_modified, note)  VALUES ('".$master_id."', 127:
now(), now(), '".$_POST["note"]."')"; 128:
$add_notes_res = mysqli_query($mysqli, $add_notes_sql)
129:                           or
die(mysqli_error($mysqli)); 130:     } 131:
mysqli_close($mysqli); 132:     $display_block = "<p>Your
entry has been added. 133:     Would you like to <a
href=\"addentry.php\">add another</a>?</p>"; 134: } 135:
?> 136: <html> 137: <head> 138: <title>Add an
Entry</title> 139: </head> 140: <body> 141: <h1>Add an
Entry</h1> 142: <?php echo $display_block; ?> 143: </body>
144: </html>
```

record addition form.


Add an Entry - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/addentry.php
Go

Add an Entry

First/Last Names:

Address:

City/State/Zip:

Address Type:
[✓ home] work other

Telephone Number:
[✓ home] work other

Fax Number:
[✓ home] work other

Email Address:
[✓ home] work other

Personal Note:

Add Entry