WEB APPLICATIONS DEVELOPMENT USING PHP & MYSQL

Course 7A

Abstract

PHP and MYSQL has been the main web development tool for it is free and open source. The authors have discussed the environmental issues in development process based on PHP and MYSQL and the implementation process of the website.

P Veera Venkata Durga PraSad

Department of Computer Science (AWDC KKD)

UNIT I

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- II

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- III

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- IV

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- V

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 - I

1) DEFINE PHP? EXPLAIN ABOUT ADVANTAGES OR BENEFITS OF PHP?

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 recursive acronym for "PHP: Hypertext Preprocessor". (OR) Personalized Home Pages

PHP is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, (like MySQL, Oracle SQL Server) session tracking, even build entire e-commerce sites.

Uses of PHP

- PHP performs system functions, i.e., from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

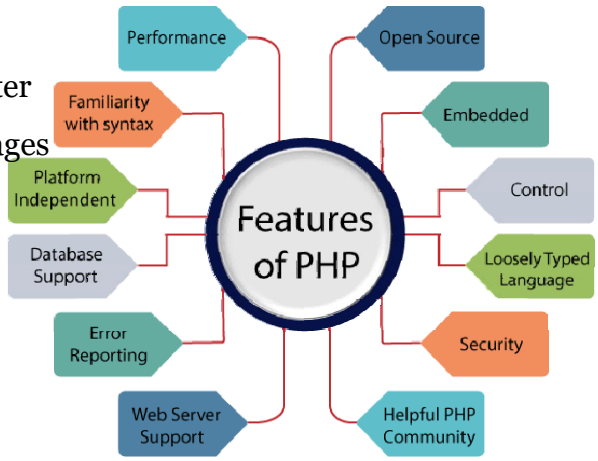
A PHP scripting block always starts with <?php and ends with ?>. A PHP scripting block can be placed anywhere in the document. On servers with shorthand support enabled you can start a scripting block with <? and end with?>.

CHARACTERSTICS OF PHP

- Five important characteristics make PHP's practical nature possible:
- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

2) EXPLAIN ABOUT FEATURES OF PHP?

PHP is very popular language because of its simplicity and open source. There are some important features of PHP



PERFORMANCE:PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses own memory, so the its server workload and time is automatically loading reduced, which results in faster processing speed and better performance.

OPEN SOURCE:PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

FAMILIARITY WITH SYNTAX:PHP has easily understandable syntax. Programmers are comfortable coding with it.

EMBEDDED:PHP code can be easily embedded within HTML tags and script.

PLATFORM INDEPENDENT:PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

DATABASE SUPPORT:PHP supports all the leading databases such as MySQL, SQLite, ODBC

ERROR REPORTING: PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

LOOSELY TYPED LANGUAGE:PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

WEB SERVERS SUPPORT:PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

SECURITY:PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

CONTROL: Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

A HELPFUL PHP COMMUNITY: It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs.

3) EXPLAIN ABOUT BUILDING BLOCKS OF PHP

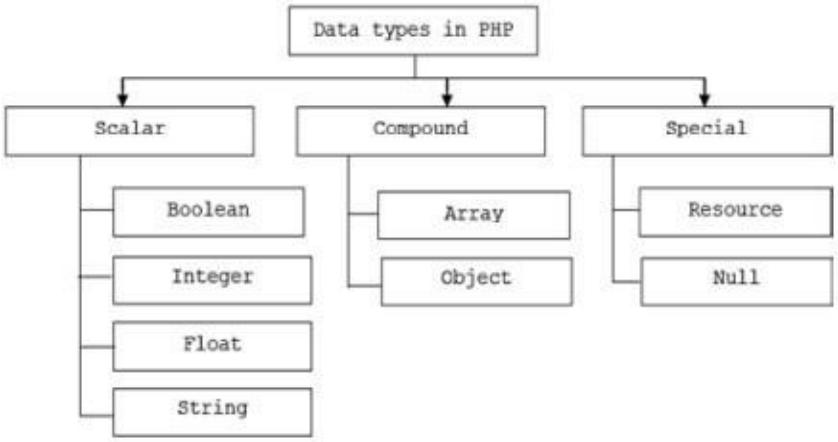
PHP's Building Blocks: They are four types of blocks will be available in the php i.e.

1) **Data Types** 2) **Literals** 3) **Variables** 4) **Constants**

DATA TYPES:

PHP supports total eight primitive data types: Integer, Floating point number or Float, String, Booleans, Array, Object, resource and NULL. These data types can be broken into three categories.

1. Scalar Data types 2. Compound Data types 3. Special Data types



LITERALS

PHP supports both integers and floating-point numbers and strings

- **Integers**— Integers are whole numbers and do not contain a decimal point for example, 123 and –6.
- **Floating-point numbers**— Floating-point numbers, also called doubles or reals, are fractional numbers such as 123.56 or –2.5.

String literals are a row of characters enclosed in either double or single quotes

- **Single quotes:** A single-quoted string does not have variables within it interpreted. Unlike the other options, variables in a single-quoted string won't be expanded when they occur. That means you need to use concatenation.
- **Double** quotes: With double quotes, the PHP compiler will expand every escaped variable inside a string literal.

VARIABLES

The way to store information in the PHP program is by using a variable. It is a name storage location capable of containing data that can be modified during program execution.

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 andunderscores (A-z, 0-9, and _).
- A variable name cannot contain spaces.
- Ex. $num=10;

CONSTANTS

A constant is an identifier (name) for a simple value. A constant value cannot change during the execution of the script. Constants are useful for storing data that doesn't change while the script is running

Constants are defined using define() function, which accepts two arguments: the name of the constant, and its value.

4) DEFINE PHP VARIABLE? EXPLAIN IT TYPES?

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

A variable starts with the $ sign, followed by the name of the variable A variable name must start with a letter or the underscore character A variable name cannot start with a number

A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _) Variable names are case-sensitive ($age and $AGE are two different variables).

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used. PHP has three different variable scopes:

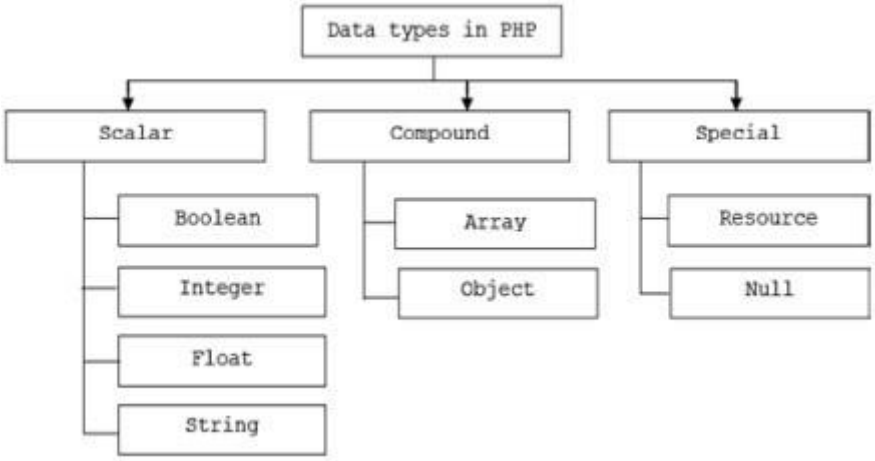
- Local
- Global
- Static

5) EXPLAIN ABOUT DATA TYPES IN PHP?

PHP supports total eight primitive data types: Integer, Floating point number or

Float, String, Booleans, Array, Object, resource and NULL. These data types can be broken into three categories

1. Scalar Data types 2. Compound Data types 3. Special Data types



1 **Scalar data types:** Scalar data types are capable of containing a single item of information.

Several data types fall under this category including Boolean, Integer, Float and String

- **Boolean:** A Boolean variable represents truth, supporting only two values TRUE or FALSE. Alternatively we can use zero to represent FALSE & nonzero to represent TRUE

- **Integer:** An integer is quite simply a whole number or that doesn't contain fractional parts.

Decimal, Octal and hexadecimal numbers

```
<?php
   $dec1 = 34;
   $oct1 = 0243;
   $hexa1 = 0x45;
   echo "Decimal number: " .$dec1. "</br>";
   echo "Octal number: " .$oct1. "</br>";
   echo "HexaDecimal number: " .$hexa1. "</br>";

?>
```

Output:

```
Decimal number: 34
Octal number: 163
HexaDecimal number: 69
```

- **Float:** Float data type allows you to specify numbers that contain fractional parts
- **String:** A string is a sequence of characters treated as a single unit; such units are typically delimited by single or double quotes.

```php
<?php
$college = "ADITYA DEGREE COLLEGE FOR WOMEN";
$address = 'Kakinada';

echo $college;
echo "<br>";
echo $address;
?>
```

Output:

```
ADITYA DEGREE COLLEGE FOR WOMEN
Kakinada
```

2. Compound data types: Compound data types allow us to aggregate multiple items of the same type under a single name

- **Array:** 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. An array is formally defined as an indexed collection of data values

```php
<?php
   $bikes = array ("Royal Enfield", "Yamaha", "KTM");
   echo "Array Element1: $bikes[0] </br>";
   echo "Array Element2: $bikes[1] </br>";
   echo "Array Element3: $bikes[2] </br>";
?>
```

Output:

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha"
[2]=> string(3) "KTM" }
Array Element1: Royal Enfield
Array Element2: Yamaha
Array Element3: KTM
```

- **Object:** 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 serves 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.

Example:

```
<?php
    class bike {
        function model() {
            $model_name = "Royal Enfield";
            echo "Bike Model: " .$model_name;
          }
    }
    $obj = new bike();
    $obj -> model();
?>
```

Output:

```
Bike Model: Royal Enfield
```

3. SPECIAL DATA TYPES: Special data types encompass those types serving some sort of specific purpose, which makes it impossible to group them in any other type category.

Resource: PHP is often used to interact with external data sources such as databases, files &network streams. Typically this interaction takes place through "handles? EX: such functions include fopen(), mysqli_connect()

Null: The special NULL value is used to represent empty variables in PHP. A variable of

type NULL is a variable without any data

6) EXPLAIN ABOUT OPERATORS IN PHP?

Operators: An operator is a symbol that specifies a particular action in an expression.

The following are the different classes of operators in PHP:

- 1. Arithmetic Operators
- 2. Assignment Operators
- 3. String Operators
- 4. Increment & Decrement Operators
- 5. Logical Operators
- 6. Equality Operators
- 7. Comparison Operators
- 8. Bit-wise Operators

1. **Arithmetic Operators:** The arithmetic operators perform various mathematical operations and will probably be used frequently in many of your PHP programs

S. No	Operator	Label	Example	Outcome
1	+	Addition	$a + $b	Sum of $a and $b
2	-	Subtraction	$a - $b	Difference of $a and $b
3	*	Multiplication	$a * $b	Product of $a and $b
4	/	Division	$a / $b	Quotient of $a and $b
5	%	Modulus	$a % $b	Remainder od $a and $b

2. Assignment Operators: The assignment operators assign a data value to a variable. The simplest form of assignment operator just assigns some value

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

3. String Operators: PHP's string operators provide a convenient way in which to concatenate strings together.

S.No	Operator	Label	Example	Outcome
1	.	Concatenation	$a="abc"."def"	$a equals "abcdef"
2	.=	Concatenation Assignment	$a .= "ghi"	$a equals its current value concatenated with "ghi"

4. Increment & Decrement Operators: Providing shortened means by which you can add 1 to or subtract 1 from the current value of a variable.

S. No	Operator	Label	Example	Outcome
1	++	Increment	$a++, ++$a	Increment $a by 1
2	--	Decrement	$a--, --$a	Decrement $a by 1

5. Logical Operators: Logical operators make it possible to direct the flow of a program, and are used frequently with control structures, such as the if conditional and the while and for loops.

S.No	Operator	Label	Example	Outcome
1	&& AND	Logical AND	$a && $b $a AND $b	True if both $a and $b are true
2	|| OR	Logical OR	$a || $b $a OR $b	True if either $a or $b is true
3	! NOT	Logical NOT	!$a NOT $a	True if $a is not true
4	XOR	Exclusive OR	$a XOR $b	True if only $a or only $b is true

6. Equality Operators: Equality operators are used to compare two values, testing for equivalence.

S.No	Operator	Label	Example	Outcome
1	= =	Is equal to	$a == $b	True if $a and $b are equivalent
2	! =	Is not equal to	$a != $b	True if $a is not equal to $b
3	= = =	Is identical to	$a ===$b	True if $a and $b are equivalent and $a & $b are of same type

7. Comparison Operators: Comparison operators, like logical operators, provide a method by which to direct program flow through examination of the comparative values of two or more variables.

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

8. Bit-wise Operators: Bitwise operators examine and manipulate integer valueson the level of individual bits that make up the integer value.

7) DEFINE CONSTANTS? EXPLAIN ABOUT TYPES OF CONSTANT AND HOW IT CAN DEFINE?

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

1. Using define() function

2. Using const keyword

Constants are similar to the variable except once they defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

PHP constant: define()

Use the define() function to create a constant. It defines constant at run time. Let's see the syntax of define() function in PHP.

To create a constant, use the define() function.

Syntax: define(name, value, case-insensitive) **name**: It specifies the constant name.

value: It specifies the constant value.

case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default

Example

```
<?php
define("MESSAGE","HelloPHP");

echo MESSAGE;
?>
```

Output:

```
Hello php
```

PHP constant: const keyword

PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

Example

```
<?php
const MESSAGE="Hello STUDENTS";

 echo MESSAGE;
?>
```

Output:

```
Hello STUDENTS
```

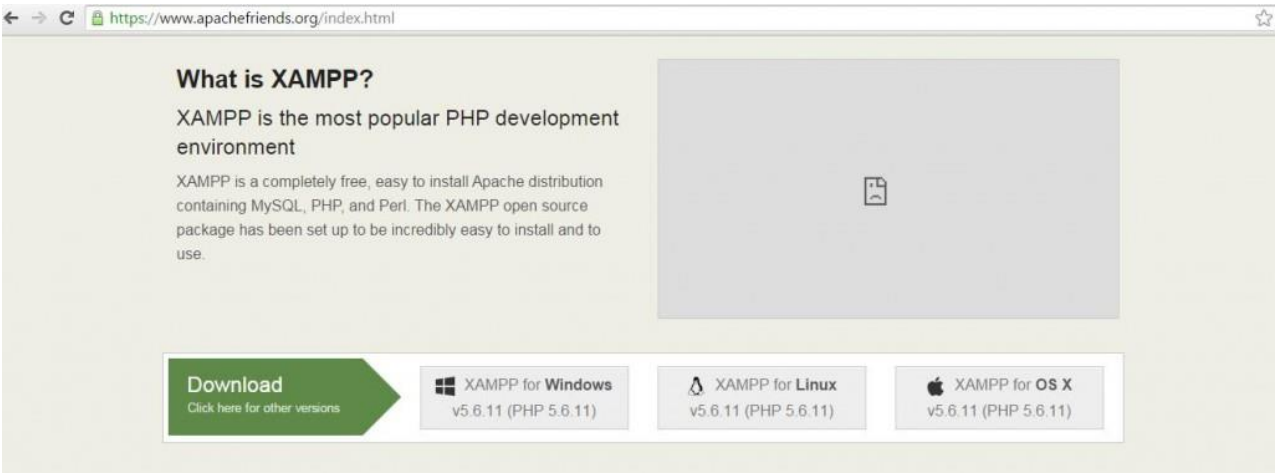
8) How to install PHP?

Before you start using PHP, you need a web host with PHP and MYSQL. For this, you should also install a web server such as Apache. To do it locally on your PC, you may download XAMPP directly from Apache Friends.

Installation of Apache, PHP, MySQL, and PHPMyAdmin

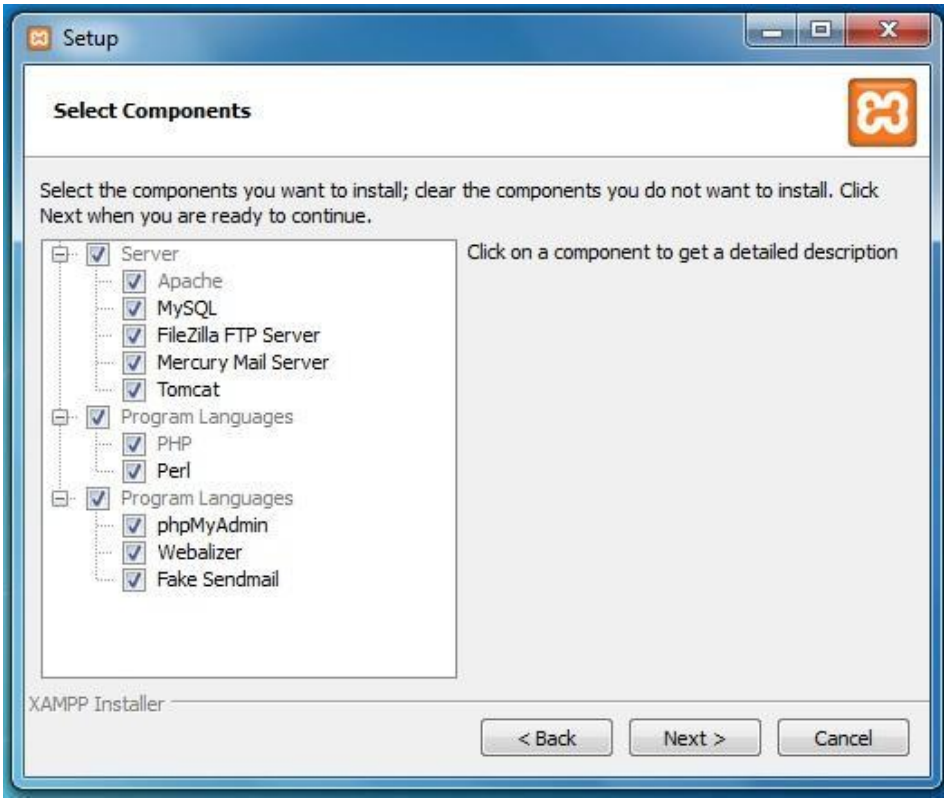
In order to start PHP, MySQL, PHPMyAdmin and Apache in a single attempt, XAMPP should be installed.

Scroll over to XAMPP for Windows and download should begin shortly.



Click the *.exe* file to start the installation procedure.

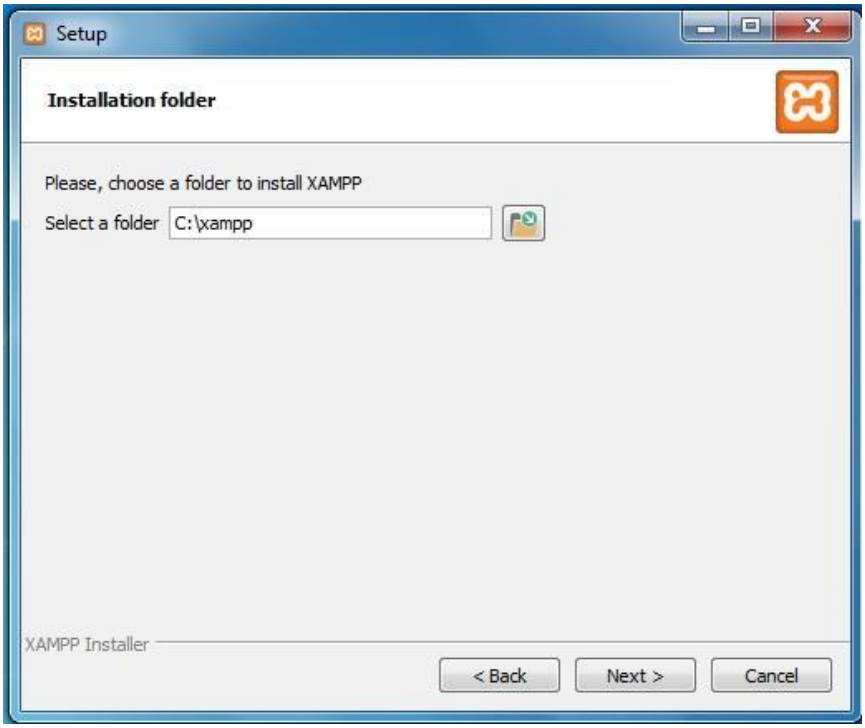
Select the components which you want to install and click “Next”.



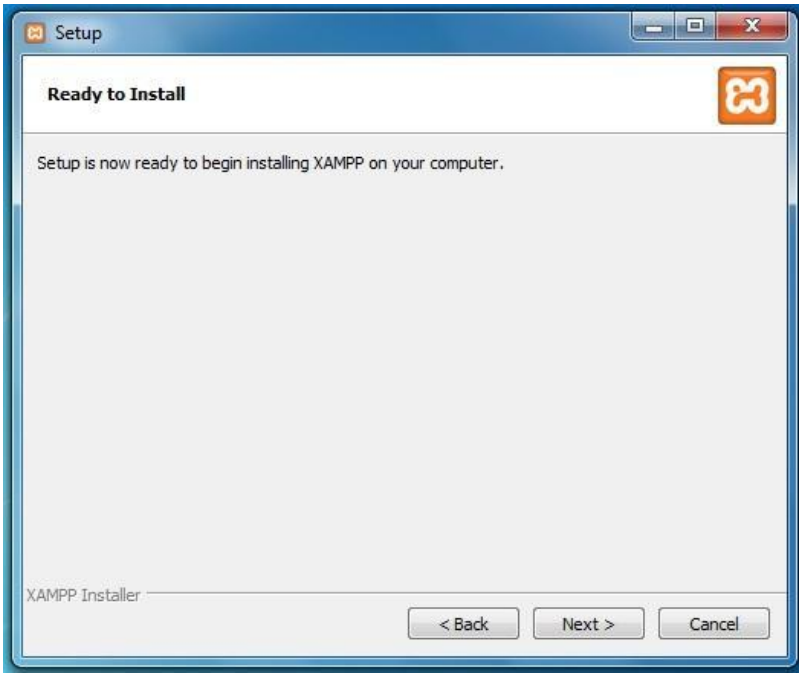
In the components area, you can view several options.

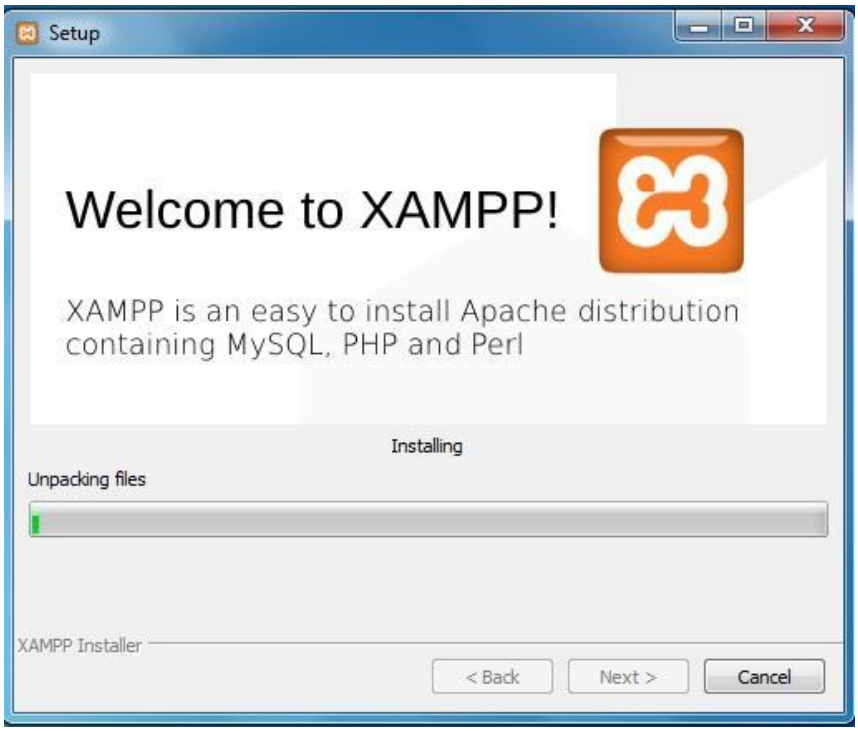
Steps:

Select your desired location, where you want to install XAMPP and then click “Next”.

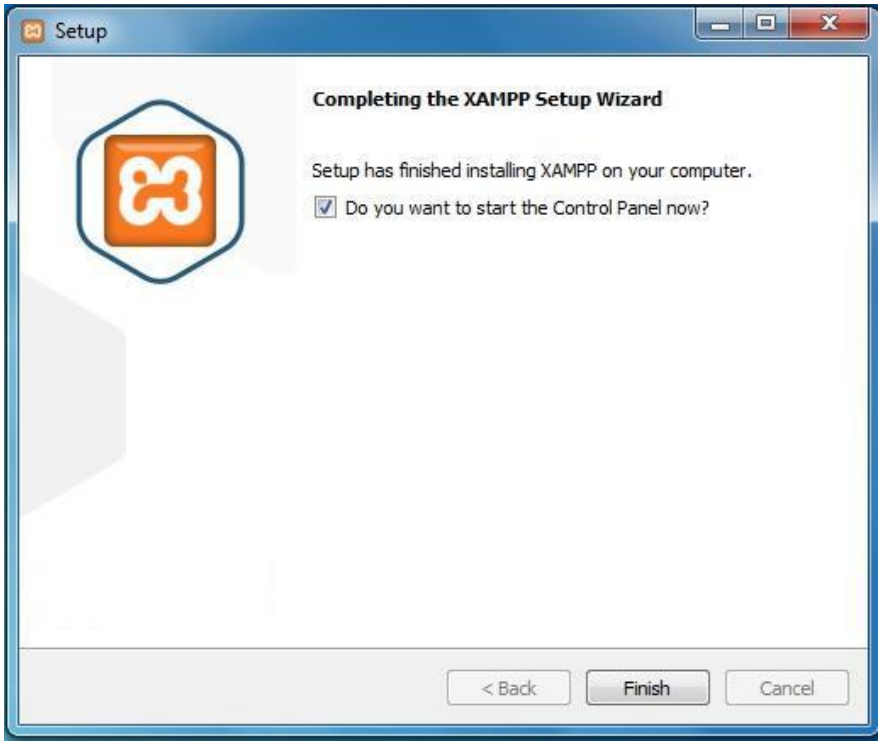


Click “Next” on the coming screens to proceed with the installation process.

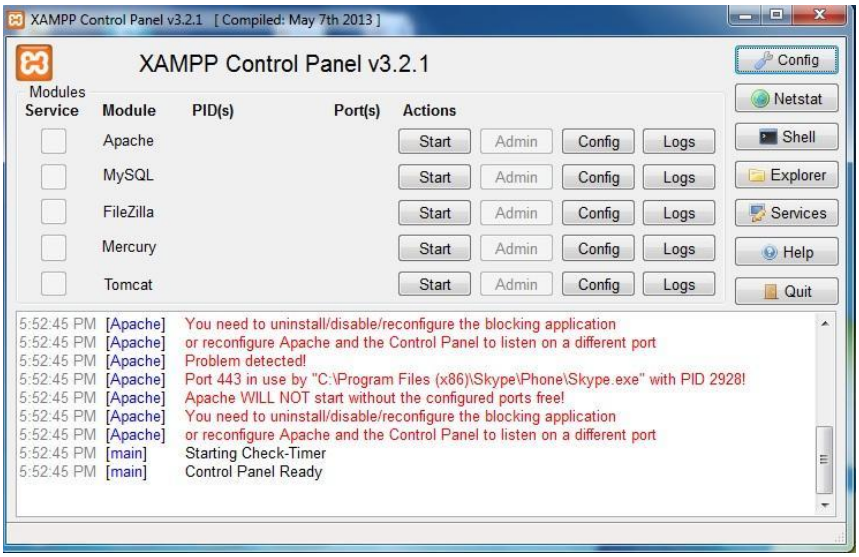


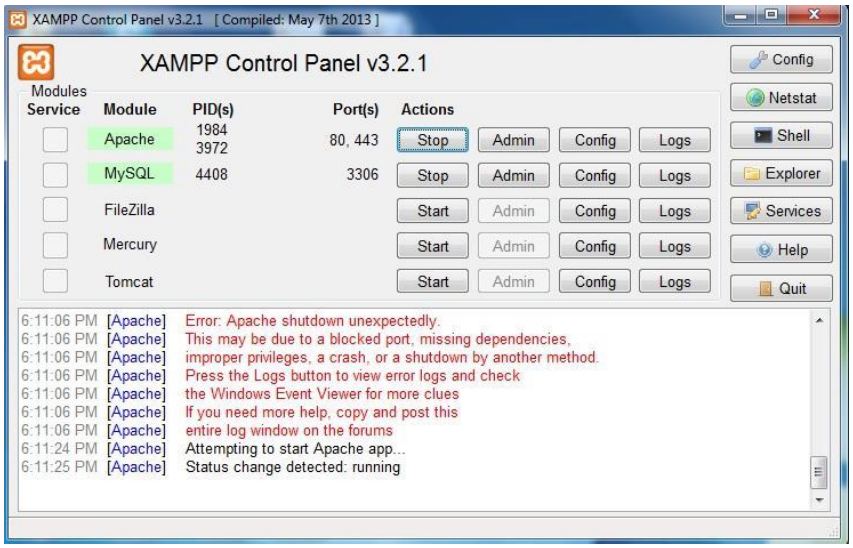


Now, you will see the final screen. I would suggest that you keep the "start the Control Panel" option checked. Click "Finish" to complete the installation process. A new window will open shortly.



The XAMPP Control Panel has now started. Now, click “Start” button in Apache and MySQL rows to begin.



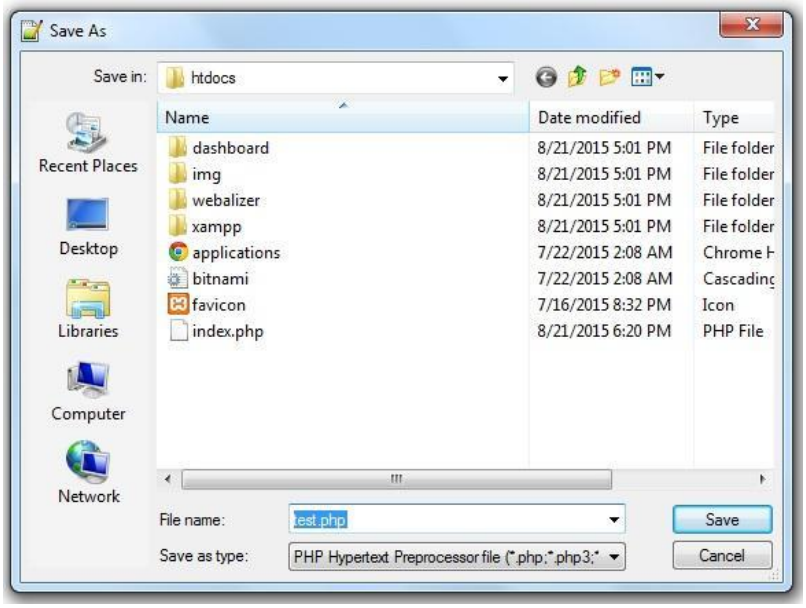


You are now ready to start writing the code. Now all you need is an editor like Notepad++ or Dreamweaver to write the code.

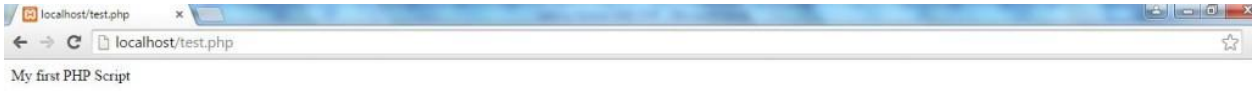
After downloading Notepad++, you can start writing your code

```
<?php
echo “My first PHP Script”;
?>
```

Now, save the page as “test.php” in *htdocs* folder and click “Save” button.



Now, open a web browser and type *localhost* in the address bar. It will automatically open the index file but if you type *localhost/test.php*, it will open the page that we have saved.



9) 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.

```
<html lang="en">
<head>
<title>A Simple PHP File</title>
</head><body>
```

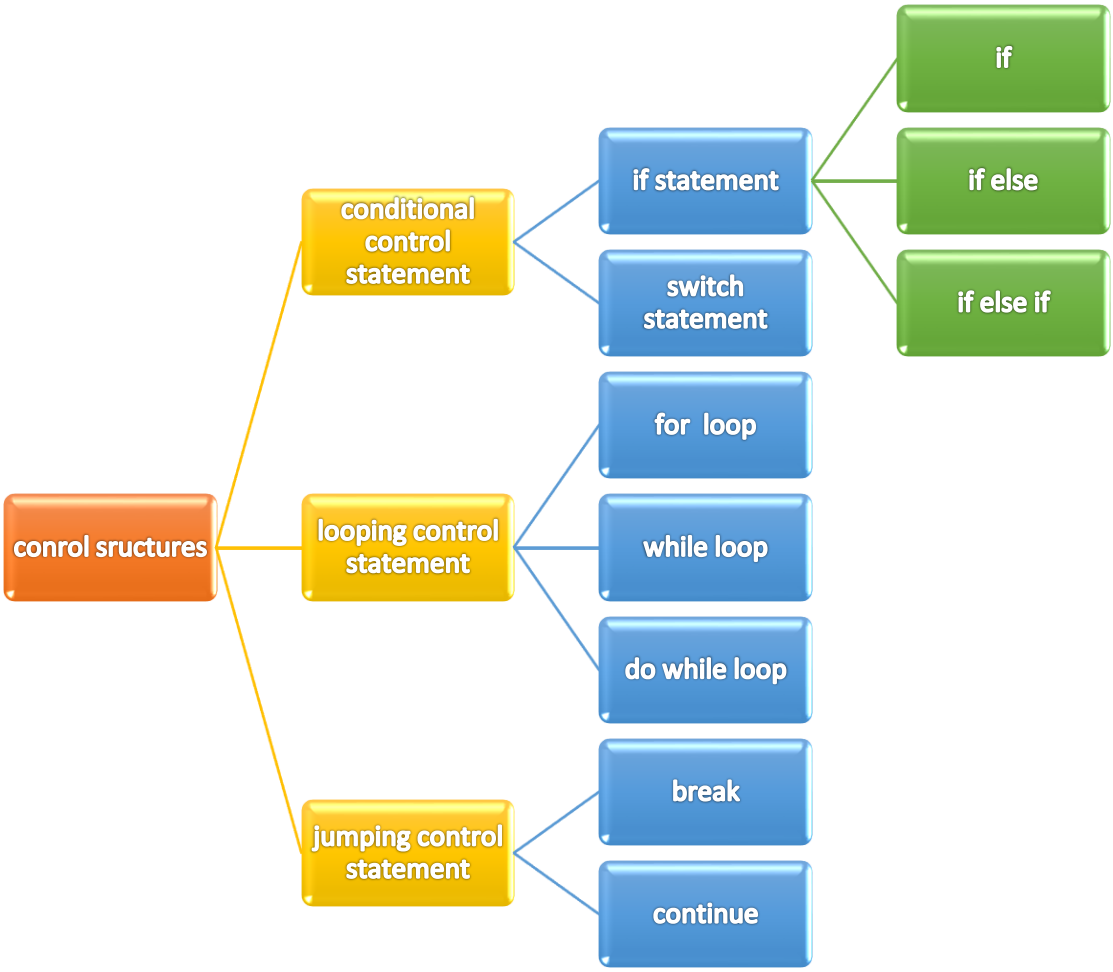
```
<h1><?php echo "Hello, world!"; ?></h1>
</body></html>
```

10) DIFFERENCE BETWEEN CONSTANTS AND VARIABLES?

A. Once the constant is defined, it can never be redefined.
B. A constant can only be defined using define() function. It cannot be defined by any simple assignment.
C. There is no need to use the dollar ($) sign before constant during the assignment.
D. Constants do not follow any variable scoping rules, and they can be defined and accessed anywhere.
E. Constants are the variables whose values can't be changed throughout the program.
A. A variable can be undefined as well as redefined easily.
B. A variable can be defined by simple assignment (=) operator.
C. To declare a variable, always use the dollar ($) sign before the variable.
D. Variables can be declared anywhere in the program, but they follow variable scoping rules.
E. The value of the variable can be changed.

11) Explain about Control Structures in PHP

The Control Structures (or) Statements used to alter the execution process of a program.



These statements are mainly categorized into following:

- Conditional Statements
- Loop Statements
- Jump Statements

Conditional Statements :

Conditional Statements performs different computations or actions depending on conditions. In PHP, the following are conditional statements

- **if** statement
- **if - else** statement
- **if - elseif - else** statement
- **switch** statement

☞ if statement

The if statement is used to test a specific condition. If the condition is true, a block of code (if-block) will be executed.

Syntax :

```
if (condtion)
{
        statements
}
```

☞ if - else statement

The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition.

Syntax :

```
if (condtion)
{
        statements
}
else
{
        statements
}
```

☞ if - elseif - else statement

The elseif statement enables us to check multiple conditions and execute the specific block of

statements depending upon the true condition among them.

Syntax :

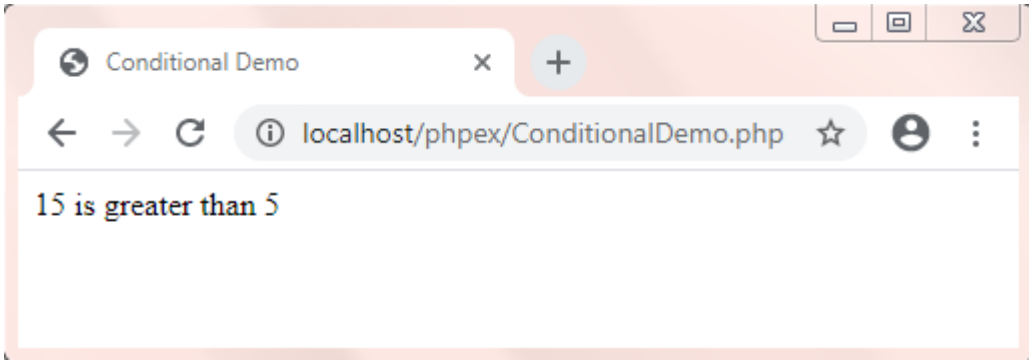
```
if (condtion1)
{
	statements
}
else if (condtion2)
{
	statements
}
else if (condtion3)
{
	statements
}
.
.
else
{
	statements
}
```

Example: "ConditionalDemo.php"

```
<html>
<head>
<title>Conditional Demo</title>
</head>
<body>
<?php
$x=15;
$y=5;
if ($x > $y)
{
    echo "$x is greater than $y";
}
else if ($x < $y)
{
    echo "$x is lessthan $y";
}
else
```

```
{
    echo "Both are Equal";
}
?>
</body>
</html>
```

Output :



☞ switch statement

The switch statement enables us to execute a block of code from multiple conditions depending upon the expression.

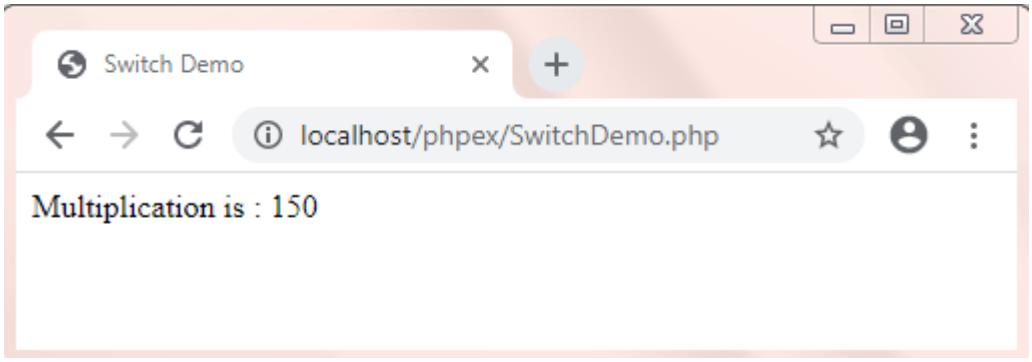
Syntax :

```
switch (expression)
{
case 1: statements
              break;
case 2: statements
              break;
case 3: statements
              break;
.
.
default: statements
}
```

Example: "SwitchDemo.php"

```
<html>
<head>
<title>Switch Demo</title>
</head>
<body>
<?php
    $x=15;
    $y=10;
    $op='*';
    switch($op)
    {
        case '+':   $z = $x + $y;
                    echo "Addition is : $z";
                    break;
        case '-':   $z = $x - $y;
                    echo "Subtraction is : $z";
                    break;
        case '*':  $z = $x * $y;
                    echo "Multiplication is : $z";
                    break;
        case '/':  $z = $x / $y;
                    echo "Division is : $z";
                    break;
        case '%':  $z = $x % $y;
                    echo "Modulus is : $z";
                    break;
        default:    echo "Invalid Operator";
    }
?>
</body>
</html>
```

Output :



Loop Statements:

Sometimes we may need to alter the flow of the program. If the execution of a specific code may need to be repeated several numbers of times then we can go for loop statements.

In PHP, the following are loop statements

- **while** loop
- **do - while** loop
- **for** loop

☞ while loop statement

With the while loop we can execute a set of statements as long as a condition is true. The while loop is mostly used in the case where the number of iterations is not known in advance.

Syntax :

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

Example: "WhileDemo.php"

```
<html>
<head>
<title>While Demo</title>
</head>
<body>
<h1>While Demo</h1>
<?php
$n=1;
while($n<=5)
{
    echo "$n <br/>";
    $n++;
}
?>
```

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

Output :

☞ do - while loop statement

The do-while loop will always execute a set of statements atleast once and then execute a set of statements as long as a condition is true.

Syntax :

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

Example: "DoWhileDemo.php"

```
<html>
<head>
<title>Do-While Demo</title>
</head>
<body>
<h1>Do-While Demo</h1>
<?php
$n=1;
do
{
```

```
    echo "$n <br/>";
    $n++;
} while($n<=5);
?>
</body>
</html>
```

Output :

☞ for loop statement

With the for loop we can execute a set of statements specified number of times. The for loop is mostly used in the case where the number of iterations is known in advance..

Syntax :

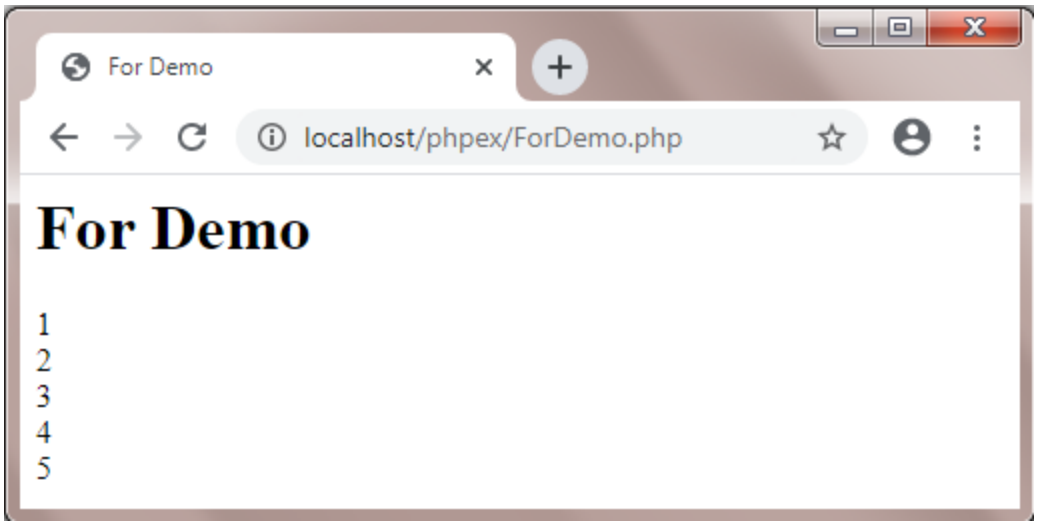
```
for (initialization; condition; increment/decrement)
{
  statements
}
```

Example: "ForDemo.php"

```
<html>
<head>
<title>For Demo</title>
</head>
<body>
```

```
<h1>For Demo</h1>
<?php
for($i=1;$i<=5;$i++)
{
    echo "$i <br/>";
}
?>
</body>
</html>
```

Output :



Jump Statements :

Jump statements in PHP are used to alter the flow of a loop like you want to skip a part of a loop or terminate a loop.

In PHP, the following are jump statements

- **break** statement
- **continue** statement

☞ break statement

The break is a keyword in php which is used to bring the program control out of the loop. i.e. when a break statement is encountered inside a loop, the loop is terminated and program control resumes at the next statement following the loop.

The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break is commonly used in the cases where we need to break the loop for a given condition.

Syntax :

```
break;
```

Example: "BreakDemo.php"

```
<html>
<head>
<title>Break Demo</title>
</head>
<body>
<h1>Break Demo</h1>
<?php
for($i=1;$i<=10;$i++)
{
    if($i==5)
    {
        break;        //terminates the current loop
    }
    echo "$i <br/>";
}
echo "Loop is Over !";
?>
</body>
</html>
```

Output :

☞ continue statement

The continue statement in php is used to bring the program control to the beginning of the loop. i.e. when a continue statement is encountered inside the loop, remaining statements are skipped and loop proceeds with the next iteration.

The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.

Syntax :

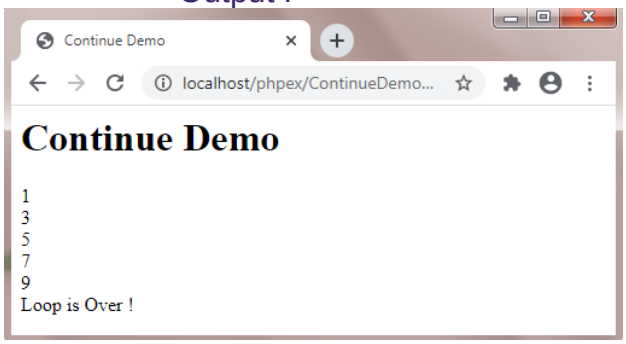
```
continue;
```

Example: "ContinueDemo.php"

```
<html>
<head>
<title>Continue Demo</title>
</head>
<body>
<h1>Continue Demo</h1>
<?php
for($i=1;$i<=10;$i++)
{
    if($i%2==0)
    {
        continue;    //terminates the current Iteration and moves to Next
```

```
    }
    echo "$i <br/>";
}
echo "Loop is Over !";
?>
</body>
</html>
```

Output :



12) EXPLAIN ABOUT CODE BLOCKS & BROWSER OUTPUT IN PHP?

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

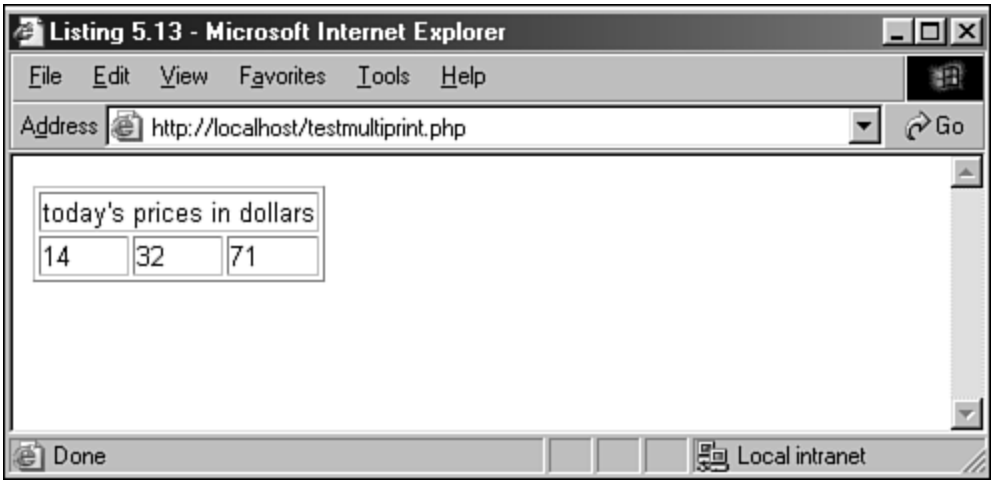
A Code Block Containing Multiple `print()` Statements

```
 1: <html>
 2: <head>
 3: <title>Listing 5.13</title>
 4: </head>
 5: <body>
 6: <?php
 7: $display_prices = true;
 8: if ($display_prices) {
 9:     print "<table border=\"1\">";
10:     print "<tr><td colspan=\"3\">";
11:     print "today's prices in dollars";
12:     print "</td></tr>";
13:     print "<tr><td>14</td><td>32</td><td>71</td></tr>";
```

```
14:     print "</table>";
15: }
16: ?>
17: </body>
18: </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 5.2.



13) EXPLAIN ABOUT ECHO AND PRINTS STATEMENTS IN PHP?

We frequently use the echo statement to display the output. There are two basic ways to get the output in PHP:

- echo
- print

echo and print are language constructs, and they never behave like a function. Therefore, there is no requirement for parentheses. However, both the statements can be used with or without parentheses. We can use these statements to output variables or strings.

14) Difference between echo and print

echo

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses.
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

```
<?php
    $fname = "Challapalli";
    $lname = "Ramu";
    echo "My name is: ".$fname,$lname;
?>
```

Output:

```
My name is: Ramu
```

print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

```
<?php
    $lang = "PHP";
    $ret = print $lang." is a web development language.";
    print "</br>";
print "Value return by print statement: ".$ret;
?>
```

Output:

```
PHP is a web development language
Value return by print statement: 1
```

15) DEFINE ABOUT FUNCTION? AND EXPLAIN ADVANTANGES OF FUNCTIONS?

functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax

```
1. function functionname(){
2. //code to be executed
3. }
```

PHP Functions Example

File: function1.php

```
1. <?php
2. function sayHello(){
3. echo "Hello PHP Function";
4. }
5. sayHello();//calling function
6. ?>
```

Output:

```
Hello PHP Function
```

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Let's see the example to pass two arguments in PHP function.

File: functionarg.php

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Ramu",27);
sayHello("Venu",29);
?>
```

Output:

```
Hello Ramu, you are 27 years old
Hello Venu, you are 29 years old
```

PHP Function: Returning Value

Let's see an example of PHP function that returns value.

File: functiondefaultarg.php

```
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

```
Cube of 3 is: 27
```

16) **Explain about Call By Value And Call By Reference In PHP?**

Call By Value

In this method, only values of actual parameters are passing to the function. So there are two addresses stored in memory. Making changes in the passing parameter does not affect the actual parameter.

Example

```
<?php
function square($num){
    $s=$num*$num;
    return $s;
}
$a = 5;
$b = square($a);
echo $a."<br>"; //Output: 5
echo $b; //Output: 25
?>
```

Output:

```
5
25
```

Call by Reference

In this method, the address of actual parameters is passing to the function. So any change made by function affects actual parameter value.

Example

```
<?php
function cube(&$num){
    $c=$num*$num*$num;
    return $c;
}
$a = 5;
```

```
$b = cube($a);
echo $a.”<br>”;
echo $b;
?>
```

Output:

```
125
125
```

.

17) Write a programs to Swapping of two numbers using call by value and call by reference?

By value:

```
<?php
function swap($arg1, $arg2){
 echo "inside function before swapping: arg1=$arg1 arg2=$arg2\n";
 $temp=$arg1;
 $arg1=$arg2;
 $arg2=$temp;
 echo "inside function after swapping: arg1=$arg1 arg2=$arg2\n";
}
$arg1=10;
$arg2=20;
echo "before calling function : arg1=$arg1
arg2=$arg2\n"; swap($arg1, $arg2);
echo "after calling function : arg1=$arg1 arg2=$arg2\n";
?>
```

OUTPUT

before calling function : arg1=10 arg2=20

inside function before swapping: arg1=10

arg2=20 inside function after swapping:

arg1=20 arg2=10 after calling function: arg1=10

arg2=20

by reference:

```
<?php
function swap(&$arg1, &$arg2){
 echo "inside function before swapping: arg1=$arg1 arg2=$arg2\n";
 $temp=$arg1;
 $arg1=$arg2;
 $arg2=$temp;
 echo "inside function after swapping: arg1=$arg1 arg2=$arg2\n";
}
$arg1=10;
$arg2=20;
echo "before calling function : arg1=$arg1
arg2=$arg2\n"; swap($arg1, $arg2);
echo "after calling function : arg1=$arg1 arg2=$arg2\n";
?
```

OUTPUT

before calling function: arg1=10 arg2=20

inside function before swapping: arg1=10

arg2=20 inside function after swapping:

arg1=20 arg2=10 after calling function:

arg1=20 arg2=10

18) EXPLAIN ABOUT RECRUSIVE FUNCTION IN PHP?

Recursive function is a function which calls itself again and again until the termination condition arrives.

Recursive function calls can save undue space and redundancy in programs and are useful for performing repetitive operations.

Factorial Number

```
<?php
function factorial($n)
{
   if ($n == 1)
{
      return 1;
}
else
$f= $n * factorial ($n -1);
 return $f;
}
   $a=5;
echo "factorial of $a= factorial($a);
?>
```

Output:

```
factorial of $5= 120
```

19) DEFINE VARIABLE SCOPE? WHAT ARE THE DIFFERENT SCOPES OF VARIABLES IN PHP?

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

Local variables: The variables declared within a function are called local variables to that function and have their scope only in that particular function. In simple words, it cannot be accessed outside that function. Any declaration of a variable outside the function with the same name as that of the one within the function is a completely different variable. For now, consider a function as a block of statements.

Example:

```
<?php

function sum($n1,$n2) {
$s=0; //local variable
$s=$n1+$n2;
return $s;
}
$a=50;
$b=30;
echo "sum of $a and $b is sum($a,$b)";

?>
```

Output:

```
Sum of 50 and 30 is 80
```

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

Example:

```
<?php
 $s=0;   //global variable
function mul($n1,$n2) {
global $s; //global variable access using global keyword
$s=$n1*$n2;
return $s;
}
$a=50;
$b=30;
echo "mul of $a and $b is mul($a,$b)";

?>
```

Output:

```
mul of 50 and 30 is 150
```

Static variable: The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

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

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

Output:

```
1
2
3
```

UNIT-2

➢ Arrays in PHP

Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key. An array is created using an **array()** function in PHP.

There are basically three types of arrays in PHP:

- **Indexed or Numeric Arrays:** An array with a numeric index where values are stored linearly.
- **Associative Arrays:** An array with a string index where instead of linear storage, each value can be assigned a specific key.
- **Multidimensional Arrays:** An array which contains single or multiple array within it and can be accessed via multiple indices.

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1st way:

```
$season=array("summer","winter","spring","autumn");
```

2nd way:

```
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
```

array1.php

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
```

```
?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

PHP Associative Array

These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="350000";
$salary["John"]="450000";
$salary["Kartik"]="200000";
```

Example

File: arrayassociative1.php

```
<?php
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000
```

PHP Multidimensional Array

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

```
$emp=array(  array(1,"sonoo",400000),  array(2,"john",500000),  array(3,"rahul",300000)  );
```

PHP Multidimensional Array Example

Let's see a simple example of PHP multidimensional array to display following tabular data. In this example, we are displaying 3 rows and 3 columns.

Id	Name	Salary
1	sonoo	400000
2	john	500000
3	rahul	300000

multiarray.php

```
<?php
$emp = array
 (
 array(1,"sonoo",400000),
 array(2,"john",500000),
 array(3,"rahul",300000)
 );
 for ($row = 0; $row < 3; $row++) {
 for ($col = 0; $col < 3; $col++) {
  echo $emp[$row][$col]."  ";
 }
 echo "<br/>";
}
```

```
?>
```

Output:

```
1 sonoo 400000
2 john 500000
3 rahul 300000
```

➢ PHP Array Functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

1) PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

2) PHP array_change_key_case() function

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

Example

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

Output:

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

3) PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

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

4) PHP count() function

PHP count() function counts all elements in an array.

Example

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

Output:

```
4
```

5) PHP sort() function

PHP sort() function sorts all the elements in an array.

Example

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

```
sort($season);

foreach( $season as $s )

{

 echo "$s<br />";

}

?>
```

Output:

```
autumn

spring

summer

winter
```

6) PHP array_reverse() function

PHP array_reverse() function returns an array containing elements in reversed order.

Example

```
<?php

$season=array("summer","winter","spring","autumn");

$reverseseason=array_reverse($season);

foreach( $reverseseason as $s )

{

 echo "$s<br />";

}

?>
```

Output:

```
autumn

spring

winter

summer
```

7) PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Example

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

Output:

```
2
```

8) PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

Example

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

Output:

```
sonoo
smith
```

➢ PHP OOP - Classes and Objects

A class is a template for objects, and an object is an instance of class.

Let's assume we have a class named Fruit. A Fruit can have properties like name, color, weight, etc. We can define variables like $name, $color, and $weight to hold the values of these properties.

When the individual objects (apple, banana, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Define a Class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

Syntax

```
<?php
class Fruit {
  // code goes here...
}
?>
```

Below we declare a class named Fruit consisting of two properties ($name and $color) and two methods set_name() and get_name() for setting and getting the $name property:

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the new keyword.

In the example below, $apple and $banana are instances of the class Fruit:

Example

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}

$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');

echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
?>
```

Apple

Banana

In the example below, we add two more methods to class Fruit, for setting and getting the $color property:

Example

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
  function set_color($color) {
    $this->color = $color;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " . $apple->get_color();
?>
```

```
Name:Apple

Color: Red
```

➢ PHP String Functions

We have to learn about the in-built functions for string processing in PHP.

In general practice, using the right string function will save you a lot of time as they are pre-defined in PHP libraries and all you have to do is call them to use them.

Below we have a list of some commonly used string functions in PHP:

strlen($str)

This function returns the length of the string or the number of characters in the string including whitespaces.

```
<?php
$str = "Welcome to Studytonight";
echo "Length of the string is: ". strlen($str);
?>
```

Length of the string is: 23

str_word_count($str)

This function returns the number of words in the string. This function comes in handly in form field validation for some simple validations.

```
<?php
$str = "Welcome to Studytonight";
echo "Number of words in the string are: ". str_word_count($str);
?>
```

Number of words in the string are: 3

strrev($str)

This function is used to reverse a string.

Let's take an example and see,

```
<?php
$str = "Welcome to Studytonight";
echo "Reverse: ". strrev($str);
?>
```

Copy

Reverse: thginotydutS ot emocleW

strpos($str, $text)

This function is used to find the position of any text/word in a given string. Just like an array, string also assign index value to the characters stored in it, starting from zero.

```
<?php
$str = "Welcome to Studytonight";
```

```
echo "Position of 'Studytonight' in string: ". strpos($str, 'Studytonight');
?>
```

Position of 'Studytonight' in string: 11

str_replace($replacethis, $replacewith, $str)

This function is used to replace a part of the string with some text. While using this function, the first argument is the part of string that you want to replace, second argument is the new text you want to include, and the last argument is the string variable itself.

Let's take an example,

```
<?php
$str = str_replace("Studytonight", "Studytonight.com", "Welcome to Studytonight");
echo $str;
?>
```

Welcome to Studytonight.com

ucwords($str)

This function is used for formatting the string. This function converts first letter/character of every word in the string to uppercase.

Let's take an example and see,

```
<?php
$str = "welcome to studytonight";
echo ucwords($str);
?>
```

Welcome To Studytonight

strtoupper($str)

To convert every letter/character of every word of the string to uppercase, one can use strtoupper() method.

```
<?php
$str = "welcome to studytonight";
echo strtoupper($str);

?>
```

WELCOME TO STUDYTONIGHT

strtolower($str)

This function is used to convert every letter/character of a string to lowercase.

```
<?php
$str = "WELCOME TO STUDYTONIGHT";
echo strtolower($str);
?>
```

welcome to studytonight

str_repeat($str, $counter)

This function is used to repeat a string a given number of times. The first argument is the string and the second argument is the number of times the string should be repeated.

```
<?php
$str = "Studytonight";
echo str_repeat($str, 4);
?>
```

StudytonightStudytonightStudytonightStudytonight

strcmp($str1, $str2)

This function is used to compare two strings. The comparison is done alphabetically. If the first string is greater than second string, the result will be greater than 0, if the first string is equal to the second string, the result will be equal to 0 and if the second string is greater than the first string, then the result will be less than 0.

```
<?php
$str1 = "Studytonight";
$str2 = "Studytonight.com";
// comparing str1 and str2
echo strcmp($str1, $str2);
// comparing str2 and str1
echo strcmp($str2, $str1);
// comparing str1 with str1
echo strcmp($str1, $str1);
?>
```

```
-4
4
0
```

substr($str, $start, $length)

This function is used to take out a part of the string(substring), starting from a particular position, of a particular length.

The first argument is the string itself, second argument is the starting index of the substring to be exracted and the third argument is the length of the substring to be extracted.

```
<?php
$str = "Welcome to Studytonight";
echo substr($str, 11, 12);
?>
```

```
Studytonight
```

trim($str, charlist)

This function is used to remove extra whitespaces from beginning and the end of a string. The second argument charlist is optional. We can provide a list of character, just like a string, as the second argument, to trim/remove those characters from the main string.

```
<?php
$str1 = "   Hello World   ";
echo trim($str1) . "<br/>";
$str2 = "Hello Hello";
echo trim($str2,"Heo");
?>
```

```
Hello World
llo Hell
```

As you can see in the output, additional spaces from the beginning and end are removed and in the second case, the characters specified are removed from the beginning and the end of the string.

explode(separator, $str, $limit)

This function is used to break a string, create an array of the broken parts of the string and return the array. The first argument, **separator** defines where to break the string from. It can be a space, hiphen(-) or any other character.

The second argument of this function is the string itself and the third argument is the **limit**, which specifies the number of array elements to return. the third argument is optional.

Let's have an example,

```
<?php


$str = "Its a beautiful day";
   print_r(explode(" ", $str));
?>
```

```
Array (
   [0] => Its
   [1] => a
   [2] => beautiful
   [3] => day
)
```

In the example above, we have provided **space** as separator to break the string and return an array.

If we provide the third argument **limit** as well, we can limit the number of array elements returned. For example, if we provide 2 as the third argument, then we will only get 2 elements in the array, the first two.

implode(separator, $arr)

This function is used to form a string using the array elements from the array provided and join them using the **separator**.

Let's take an example,

```
<?php
$arr = array("Its", "a", "beautiful", "day");
// <br> is used to jump to next line
echo implode(" ", $arr) . "<br>";
echo implode("-", $arr) . "<br>";
echo implode("/", $arr) . "<br>";
?>
```

```
Its a beautiful day
Its-a-beautiful-day
```

```
Its/a/beautiful/day
```

➢ PHP Date and Time

- The date & time using the date() function in PHP, we will also see the various formatting options available with these functions & understand their implementation through the examples.

Date and time are some of the most frequently used operations in PHP while executing SQL queries or designing a website etc. PHP serves us with predefined functions for these tasks. Some of the predefined functions in PHP for date and time are discussed below.

PHP date() Function: The PHP date() function converts timestamp to a more readable date and time format.

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 epoch (midnight Greenwich Mean Time on January 1, 1970, i.e. January 1, 1970, 00:00:00 GMT). Since this is an impractical format for humans to read, PHP converts timestamp to a format that is readable and more understandable to humans.

Syntax:

date(format, timestamp)

Explanation:

- The format parameter in the date() function specifies the format of returned date and time.
- The timestamp is an optional parameter, if it is not included then the current date and time will be used.

Example: The below program explains the usage of the date() function in PHP.

```
<?php
 echo "Today's date is :";
 $today = date("d/m/Y");
 echo $today;
?>
```

Output:

```
Today's date is :05/12/2017
```

Formatting options available in date() function:

The format parameter of the date() function is a string that can contain multiple characters allowing to generate the dates in various formats. Date-related formatting characters that are commonly used in the format string:

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

The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting.

Example: The below example explains the usage of the date() function in PHP.

```
<?php
 echo "Today's date in various formats:" . "\n";
 echo date("d/m/Y") . "\n";
 echo date("d-m-Y") . "\n";
 echo date("d.m.Y") . "\n";
 echo date("d.M.Y/D");
?>
```

Output:

```
Today's date in various formats:
05/12/2017
05-12-2017
05.12.2017
05.Dec.2017/Tue
```

The following characters can be used along with the date() function to format the time string:

- h: Represents hour in 12-hour format with leading zeros (01 to 12).
- H: Represents hour in 24-hour format with leading zeros (00 to 23).
- i: Represents minutes with leading zeros (00 to 59).
- s: Represents seconds with leading zeros (00 to 59).
- a: Represents lowercase antemeridian and post meridian (am or pm).
- A: Represents uppercase antemeridian and post meridian (AM or PM).

Example: The below example explains the usage of the date() function in PHP.

```
<?php
 echo date("h:i:s") . "\n";
 echo date("M,d,Y h:i:s A") . "\n";
 echo date("h:i a");
?>
```

Output:

```
03:04:17
Dec,05,2017 03:04:17 PM
03:04 pm
```

UNIT- 3

❖ PHP Forms

In HTML, forms are used to collect user input/information. But, as far as HTML goes, it just provides the graphical interface and the ability to write on the input fields. However, the aim we collect user information is because we need to process this information. That usually means saving the inputs to a database so that they can be accessed later for several purposes. In this section, we will not send information to the database, as that requires an active database that is already set up and includes knowledge form SQL language, but we can retrieve information that the user has given us.

That being said, it is important to have a way to get information, because what you use it for and where you want to show/save it depends. Let's have a look at the HTML form below:

```
<html>
<head>
<title>student form</title>
</head>
<body>
<form method="POST"action="myform.php">
    <input type="text"name="name"placeholder="Name"><br>
    <input type="email"name="email"placeholder="E-Mail"><br>
    <input type="number"name="age"placeholder="Age"><br>
    <input type="radio"name="gender"value="Male">Male
    <input type="radio"name="gender"value="Female">Female<br>
    <input type="submit"name="submit"value="Submit">
</form>
</body>
</html>
```

This form is just a regular one, and includes inputs for name, e-mail, age and gender. This information will be subject of a **print** when the Submit button is clicked. That just proves that we got the information from the user. For now, let's see what we got and fill the form:

```
<html>
<head>
<title> form </title>
</head>
<body>
<div class="cs">
    <h2 class="text"> student data</h2>
<?php
$name=$_POST['name'];
$email=$_POST['email'];
$age=$_POST['age'];
$gender=$_POST['gender'];
if(empty($name)) {
    echo "enter student name";
}
else if(empty($email)) {
    echo "enter email id";
}
else if(empty($age)) {
    echo "enter age";
}
else if(empty($gender)) {
    echo "select gender";
}
else
 echo"Name:".$name."<br>email:".$email."<br>age:".$age."<br>gender:".$gender;
?>
</div>
</body>
</html>
```

Next, we check each input to make sure the user has written/chosen something, and the input is not empty. We do this using two well-known functions in PHP, the empty(). After making sure we have this right, we can also validate fields. In this case, only the name input may need some sort of validation to make sure one simply doesn't write numbers in there, while other inputs are more or less validated from HTML5, in the case of email and age.

❖ ACCESSING FORM INPUT WITH USER-DEFINED ARRAYS

Now we know that how to gather information from HTML elements that submit a single value per element name, such as text fields, text areas, and radio buttons. But there is a problem when working with elements such as checkboxes because it is possible for the user to choose one or more items.

In this case we can use arrays to access the multiple data, in case of check boxes.

```
<form method="POST"action="FormHandling.php">
        <input type="text"name="name"placeholder="Name"><br><br>
       <h2>Select languages</h2><br>
        <input type="checkbox"name="language[]"value="telugu">telugu
        <input type="checkbox"name="language[]"value="english">english
        <input type="checkbox"name="language[]"value="hindi">hindi
        <input type="checkbox"name="language[]"value="urdhu">urdhu

        <br><br>

        <input type="submit"name="submit"value="Submit">

        </form>
```

Put these lines into a text file called form with formhandling.html and place that file in your web server document root.

Line 7 of the script in accesses the $_POST['name'] variable, which is derived from the user INPUT field in the form. On line 10, the code tests for the $_POST['language'] variable.

Submit the form, and you might see something like that

```
<html>
<head>
   <title>cs</title>
   <style>


      div{
         background-image: linear-gradient(to left top,rgb(255,0,0),rgb(30,0,0));
         position:relative;
         left:20px;
         top:30px;
         width:300px;
         height:150px;
         padding:50px;
         color:rgb(229, 226, 226);
```

```
            border-radius: 1rem;
            border:2px solid blueviolet;
        }
    </style>
</head>
<body>
    <div>

<?php
$name=$_POST['name'];
$lan=$_POST['language'];
if((!isset($name))||(!isset($lan))){
    echo "enter fields";
}
else
{
echo "hello ".$name."<br>";
echo "your selected languages are";
echo "<ul>";
foreach($lan as $val){
    echo "<li>$val</li>";
}
echo"</ul>";
}
?>
</html>
```

❖ Combining HTML and PHP Code on a Single Page

- Write PHP code with HTML code on a single page as on hard copy.
- More flexibility to write the entire page dynamically.
- For this use a PHP_SELF variable is in the action field of the <form> tag.
- The action field of the FORM instructs where to submit the form data when the user presses the "submit" button.
- The same PHP page as the handler for the form as well.
- The action field of form use to switch to control to other page but Using PHP_SELF variable do not need to edit the action field.

Example:-

A file called self.php and want to load the same page after the form is submitted.

<form method="post" action="self.php">

We can use the PHP_SELF variable instead of "self.php".

The code becomes:

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

Example :-

```
<?php
if(isset($_POST['submit']))
{
    $name = $_POST['name'];
    echo "Glad to meet you <b> $name </b>";
    }
?>
```

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
   <input type="text" name="name"><br>
   <input type="submit" name="submit" value="Submit">
</form>
```

❖ Hiding Fild

The <input type="hidden"> defines a hidden input field.

A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted.

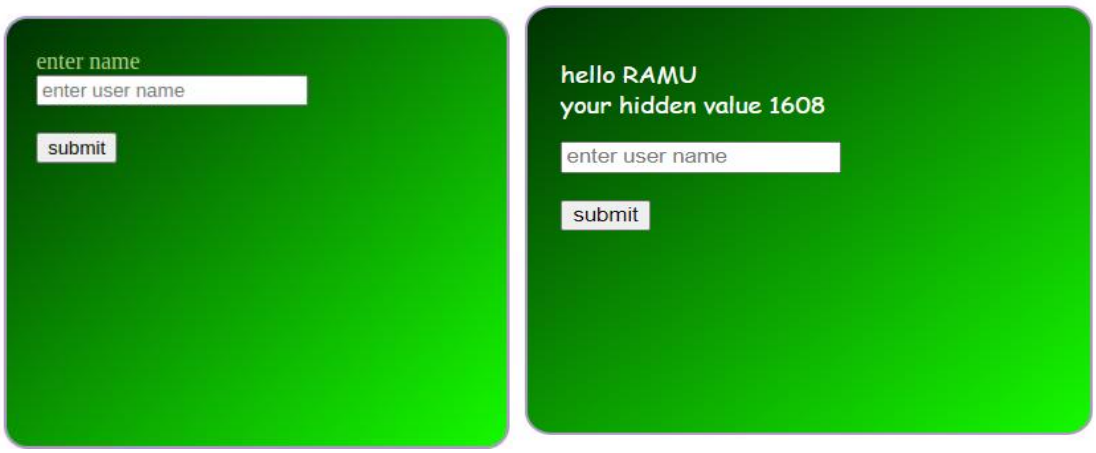
A hidden field often stores what database record that needs to be updated when the form is submitted.

While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

```
<html>
 <title>cs</title>
   <style>
      body{


      }
      div{
         background-image: linear-gradient(to left top,rgb(21,250,0),rgb(0,50, 2));
```

```
            position:relative;
            left:20px;
            top:30px;
            width:300px;
            height:250px;
            padding:20px;
            color:rgb(190, 213, 143);
            border-radius: 1rem;
            border:2px solid rgb(177, 152, 200);
        }
        P{
            font-size:15px;
            font-family:cursive;
            color:#fff;
        }
    </style>
</head>
<body>
    <div>
<?php
if(empty($_POST['name'])){
    echo "enter name";
}
else
{
    $name=$_POST['name'];
    echo "<p>hello ".$name."<br>";
    $hidden=$_POST['hid'];
    echo "your hidden value ".$hidden."</p>";
}
?>
    <form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

```
    <input type="text" name="name" placeholder="enter user name"><br>
    <input type="hidden" name="hid" value="1608"><br>
    <input type="submit" value="submit">
</form>
</body>
</html>
```



❖ Redirection in PHP

The header function in PHP can be used to redirect the user from one page to another. It is an in-built function that sends raw HTTP header to the destination (client).

Syntax of header function

header($header_value, $replace_value, $http_response_code)

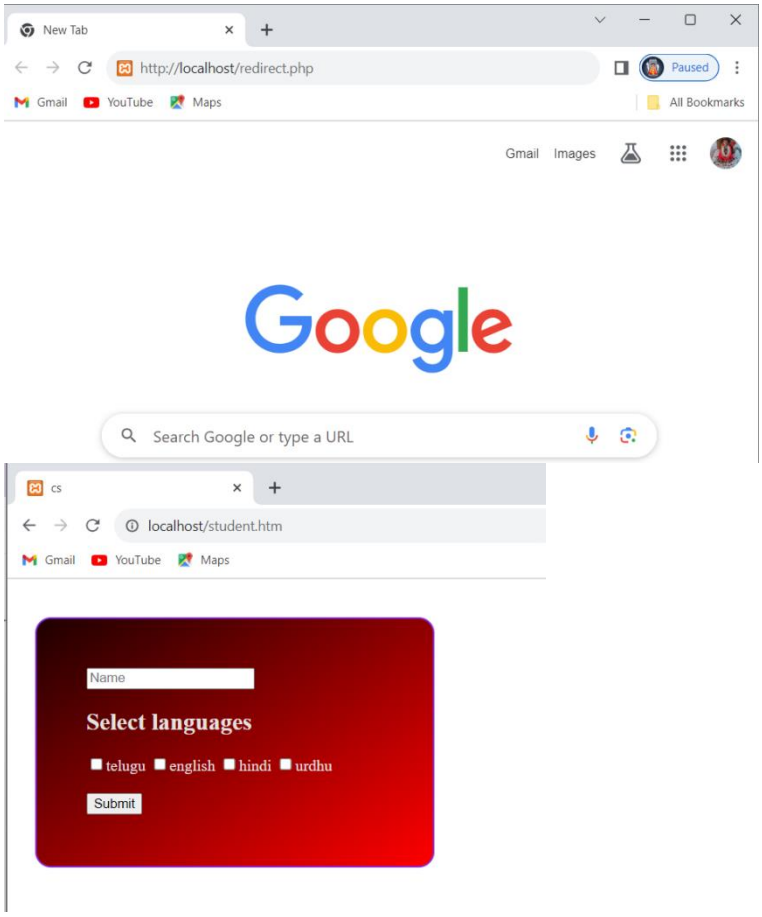
Following are the parameters –

- The 'header_value' in the function is used to store the header string.
- The 'replace_value' parameter stores the value that needs to be replaced.
- The 'response_code' is used to store the HTTP response code.
-

Example

```
<?php
  header("Location: student.html");
  exit;
?>
```

The website to which the page needs to be redirected is specified in the header function.



❖ PHP mail

PHP mail is the built in PHP function that is used to send emails from PHP scripts.

The mail function accepts the following parameters:

- Email address
- Subject
- Message
- CC or BC email addresses
- It's a cost effective way of notifying users on important events.
- Let users contact you via email by providing a contact us form on the website that emails the provided content.
- Developers can use it to receive system errors by email
- You can use it to email your newsletter subscribers.
- You can use it to send password reset links to users who forget their passwords
- You can use it to email activation/confirmation links. This is useful when registering users and verifying their email addresses

use the mail PHP

Sending mail using PHP

The PHP mail function has the following basic syntax

```
<?php
mail($to_email_address,$subject,$message,[$headers],[$parameters]);
?>
```

HERE,

- "$to_email_address" is the email address of the mail recipient
- "$subject" is the email subject
- "$message" is the message to be sent.
- "[$headers]" is optional, it can be used to include information such as CC, BCC
    - CC is the acronym for carbon copy. It's used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.
    - BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.

```
<?php

$receiver = "aditya16@gmail.com";

$subject = "Email Test via PHP using Localhost";

$body = "Hi, there...This is a test email send from Localhost.";

$sender = "From:awdckkd@gmail.com";



if(mail($receiver, $subject, $body, $sender)){

    echo "Email sent successfully to $receiver";

}else{

    echo "Sorry, failed while sending mail!";

}

?>
```

❖ PHP File Upload

With PHP, it is easy to upload files to the server.

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

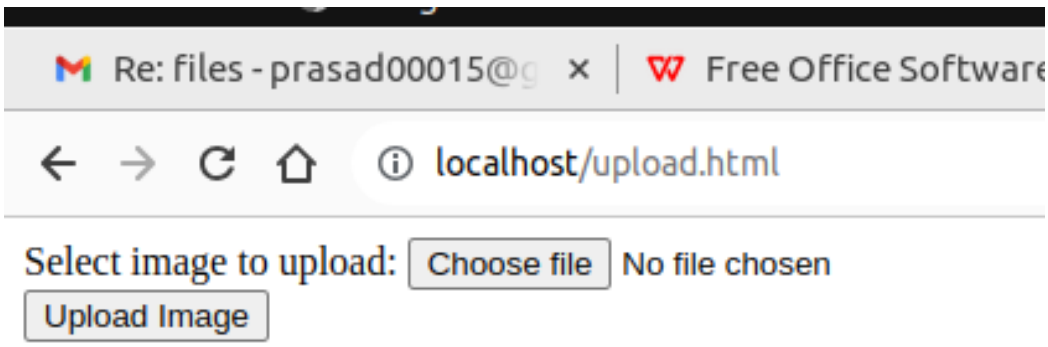
In your "php.ini" file, search for the file_uploads directive, and set it to On:

file_uploads = On

Create The HTML Form

Next, create an HTML form that allow users to choose the image file they want to upload:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
  Select image to upload:
  <input type="file" name="fileToUpload">
  <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```



Some rules to follow for the HTML form above:

- Make sure that the form uses method="post"
- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

Without the requirements above, the file upload will not work.

Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

The form above sends data to a file called "upload.php", which we will create next.

1. Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
    $uploadOk = 1;
  } else {
    echo "File is not an image.";
    $uploadOk = 0;
  }
}
?>
```

PHP script explained:

- ✓ $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- ✓ $target_file specifies the path of the file to be uploaded
- ✓ $uploadOk=1 is not used yet (will be used later)
- ✓ $imageFileType holds the file extension of the file (in lower case)
- ✓ Next, check if the image file is an actual image or a fake image

Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

2. Check if File Already Exists

Now we can add some restrictions.

First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}
```

3. Limit File Size

The file input field in our HTML form above is named "fileToUpload".

Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}
```

4. Limit File Type

The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

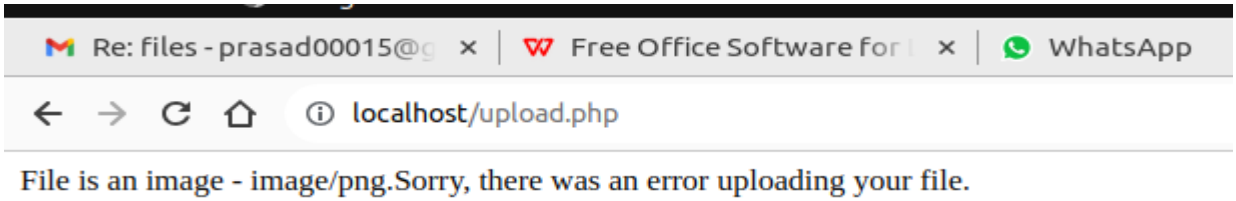
```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
  echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
  $uploadOk = 0;
}
```

Complete Upload File PHP Script

The complete "upload.php" file now looks like this:

```
<?php
$target_dir = "uploads/";
```

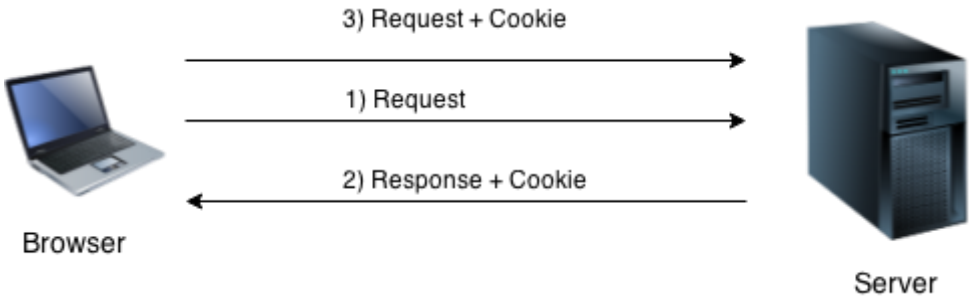
```php
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
    $uploadOk = 1;
  } else {
    echo "File is not an image.";
    $uploadOk = 0;
  }
}

// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
  echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
  $uploadOk = 0;
}

// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
  echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
    echo "The file ". htmlspecialchars( basename( $_FILES["fileToUpload"]["name"])). " has been
uploaded.";
  } else {
    echo "Sorry, there was an error uploading your file.";
  }
}
?>
```



❖ PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

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.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

```php
<?php
setcookie(mycookie,Hello students, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
```

```php
<html>
<body>

<?php
if(!isset($_COOKIE['mycookie'])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie mycookie is set!<br>";
  echo "Value is: " . $_COOKIE['mycookie'] ;
}
?>

</body>
</html>
```

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

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

Example

```php
<?php
setcookie(mycookie,Hello Bsc students, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
```

```php
<html>
<body>

<?php
if(!isset($_COOKIE['mycookie'])) {
```

```
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie mycookie is set!<br>";
  echo "Value is: " . $_COOKIE['mycookie'] ;
}
?>

</body>
</html>
```

Delete a Cookie

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

Example

```
<?php
// set the expiration date to one hour ago
setcookie(mycookie,Hello Bsc students, time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie(mycookie,Hello Bsc students, time() + (86400 * 30), "/");

?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
```

```
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>

</body>
</html>
```

❖ session in php

In general, session refers to a frame of communication between two medium. A PHP session is used to store data on a server rather than the computer of the user. Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc.

sessions better than cookies

Although cookies are also used for storing user related data, they have serious security issues because cookies are stored on the user's computer and thus they are open to attackers to easily modify the content of the cookie. Addition of harmful data by the attackers in the cookie may result in the breakdown of the application.
Apart from that cookies affect the performance of a site since cookies send the user data each time the user views a page. Every time the browser requests a URL to the server, all the cookie data for that website is automatically sent to the server within the request.

Below are different steps involved in PHP sessions:

- **1. Starting a PHP Session**:
- The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP **session_start()** function is used to begin a new session.It also creates a new session ID for the user.
  Below is the PHP code to start a new session:

- **Storing Session Data**: Session data in key-value pairs using the **$_SESSION**[] superglobal array.The stored data can be accessed during lifetime of a session.
  Below is the PHP code to store a session with two session variables Rollnumber and Name:

```
<?php

session_start();

?>
```

```php
<?php

session_start();

$_SESSION["Rollnumber"] = "11";
$_SESSION["Name"] = "Ajay";

?>
```

- **2. Accessing Session Data**: Data stored in sessions can be easily accessed by firstly calling **session_start()** and then by passing the corresponding key to the **$_SESSION** associative array.
  The PHP code to access a session data with two session variables Rollnumber and Name is shown below:

```php
<?php

session_start();

echo 'The Name of the student is :' . $_SESSION["Name"] . '<br>';
echo 'The Roll number of the student is :' . $_SESSION["Rollnumber"] . '<br>';

?>
```

Output:

The Name of the student is :Ajay

The Roll number of the student is :11

- **3. Destroying Certain Session Data**: To delete only a certain session data,the unset feature can be used with the corresponding session variable in the **$_SESSION** associative array.

The PHP code to unset only the "Rollnumber" session variable from the associative session

```
<?php

session_start();

if(isset($_SESSION["Name"])){
    unset($_SESSION["Rollnumber"]);
}

?>
```

array:

- **4. Destroying Complete Session**: The **session_destroy()** function is used to completely destroy a session. The session_destroy() function does not require any argument.

```
<?php

session_start();
session_destroy();

?>
```

Important Points

1. The session IDs are randomly generated by the PHP engine .
2. The session data is stored on the server therefore it doesn't have to be sent with every browser request.
3. The session_start() function needs to be called at the beginning of the page, before any output is generated by the script in the browser.

❖ Login Flow With Sessions and Cookies

Let's quickly go through a common login flow for a website to understand what happens behind the scenes.

1. A user opens the login page of a website.
2. After submitting the login form, a server on the other end authenticates the request by validating the credentials that were entered.
3. If the credentials entered by the user are valid, the server creates a new session. The server generates a unique random number, which is called a session id. It also creates a new file on the server which is used to store the session-specific information.
4. Next, a session id is passed back to the user, along with whatever resource was requested. Behind the scenes, this session id is sent in the `PHPSESSID` cookie in the response header.
5. When the browser receives the response from the server, it comes across the `PHPSESSID` cookie header. If cookies are allowed by the browser, it will save this `PHPSESSID` cookie, which stores the session id passed by the server.
6. For subsequent requests, the `PHPSESSID` cookie is passed back to the server. When the server comes across the `PHPSESSID` cookie, it will try to initialize a session with that session id. It does so by loading the session file which was created earlier, during session initialization. It will then initialize the super-global array variable `$_SESSION` with the data stored in the session file.

In this way, the user data is preserved across multiple requests, and the user is kept logged in throughout a session.

UNIT-IV

❖ PHP Include and Require

PHP allows us to create various elements and functions, which are used several times in many pages. It takes much time to script these functions in multiple pages. Therefore, use the concept of **file inclusion** that helps to include files in various programs and saves the effort of writing code multiple times.

"PHP allows you to include file so that a page content can be reused many times. It is very helpful to include files when you want to apply the same HTML or PHP code to multiple pages of a website." There are two ways to include file in PHP.

Files are included based on the file path given or, if none is given, the include_path specified. If the file isn't found in the include_path, include will finally check in the calling script's own directory and the current working directory before failing. The include construct will emit an **E_WARNING** if it cannot find a file; this is different behavior from require, which will emit an **E_ERROR**.

1. include
2. require

Both include and require are identical to each other, except failure.

- **include** only generates a warning, i.e., E_WARNING, and continue the execution of the script.
- **require** generates a fatal error, i.e., E_COMPILE_ERROR, and stop the execution of the script.

Advantage

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

Easy editable: If we want to change anything in webpages, edit the source file included in all webpage rather than editing in all the files separately.

PHP include

PHP include is used to include a file on the basis of given path. You may use a relative or absolute path of the file.

Syntax

There are two syntaxes available for include:

```
include 'filename ';
Or
include ('filename');
```

Examples

Let's see a simple PHP include example.

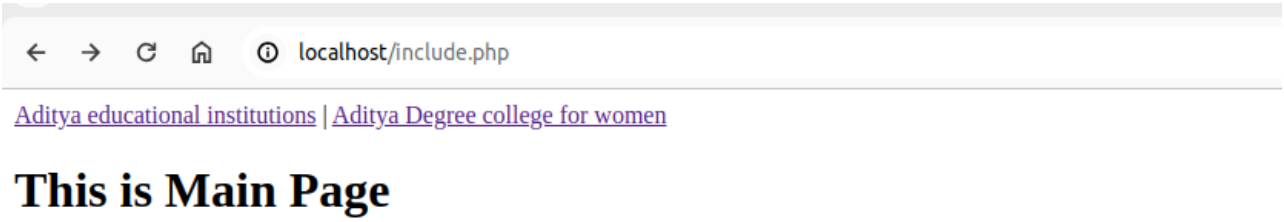
File: menu.html

```
<a href="https://aditya.ac.in/">Aditya educational institutions</a>
<a href="https://aditya.ac.in/awdckkd/">Aditya Degree college for women </a>
```

File: include.php

```
<?php include("menu.html"); ?>
<h1>This is Main Page</h1>
```

Output:



PHP require

PHP require is similar to include, which is also used to include files. The only difference is that it stops the execution of script if the file is not found whereas include doesn't.

Syntax

There are two syntaxes available for require:

```
require 'filename';
Or
require ('filename');
```

Examples

Let's see a simple PHP require example.

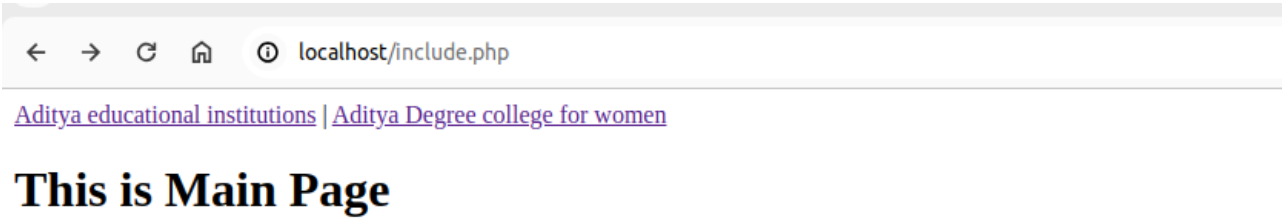
File: menu.html

```
<a href="https://aditya.ac.in/">Aditya educational institutions</a>
<a href="https://aditya.ac.in/awdckkd/">Aditya Degree college for women </a>
```

File: require.php

```
<?php require("menu.html"); ?>
<h1>This is Main Page</h1>
```

Output:



❖ PHP File() Handling & Functions

File

A file is simply a resource for storing information on a computer.

Files are usually used to store information such as:

- Configuration settings of a program
- Simple data such as contact names against the phone numbers.
- Images, Pictures, Photos, etc.

PHP File Formats Support

PHP file functions support a wide range of file formats that include:

- File.txt
- File.log
- File.custom_extension i.e. file.xyz
- File.csv
- File.gif, file.jpg etc
- Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.
- You want to store simple data such as server logs for later retrieval and analysis
- You want to store program settings i.e. program.ini

PHP file() Functions

PHP provides a convenient way of working with files via its rich collection of built in

Let's now look at some of the most commonly used PHP file functions.

1. PHP file_exists() Function

This function is used to determine whether a file exists or not.

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

The file_exist function has the following syntax.

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

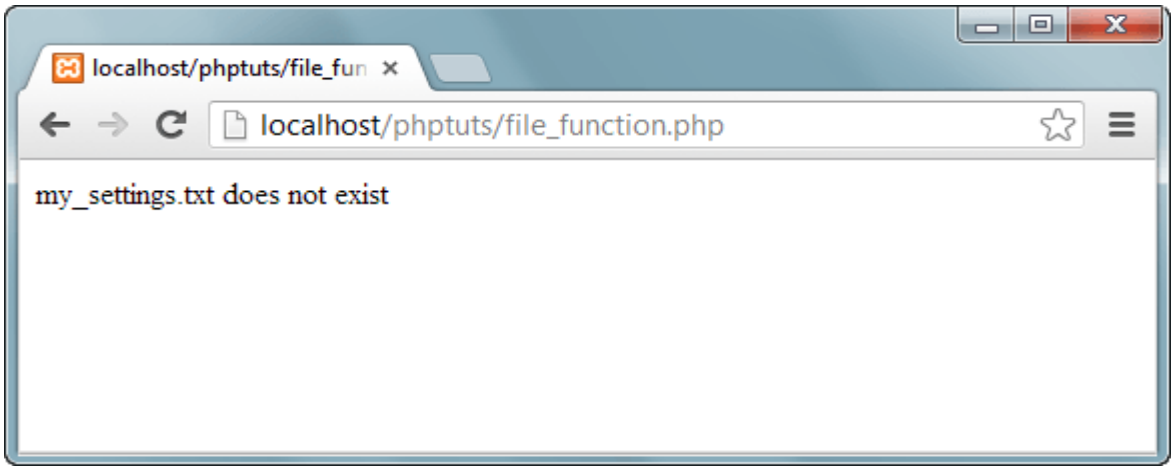
HERE,

- "file_exists()" is the PHP function that returns true if the file exists and false if it does not exist.
- "$file_name" is the path and name of the file to be checked

The code below uses file_exists function to determine if the file my_settings.txt exists.

```
<?php
if (file_exists('my_settings.txt'))
 {
      echo 'file found!';
 }
else
 {
      echo 'my_settings.txt does not exist';
  }
?>
```

Save the above code in a file named file_function.php Assuming you saved the file in phptuts folder in htdocs, open the URL **http://localhost/phptuts/file_function.php** in your browser You will get the following results.



2. PHP fopen() Function

The fopen function is used to open files. It has the following syntax

```
<?php
fopen($file_name,$mode,$use_include_path,$context);
 ?>
```

HERE,

- "fopen" is the PHP open file function
- "$file_name" is the name of the file to be opened

- "$mode" is the mode in which the file should be opened, the table below shows the modes

Mode	Description
R	• Read file from beginning. • Returns false if the file doesn't exist. • Read only
r+	• Read file from beginning • Returns false if the file doesn't exist. • Read and write
W	• Write to file at beginning • truncate file to zero length • If the file doesn't exist attempt to create it. • Write only
w+	• Write to file at beginning, truncate file to zero length • If the file doesn't exist attempt to create it. • Read and Write
A	• Append to file at end • If the file doesn't exist attempt to create it. • Write only
a+	• Php append to file at end • If the file doesn't exist attempt to create it • Read and write

- "$use_include_path" is optional, default is false, if set to true, the function searches in the include path too.
- "$context" is optional, can be used to specify the context support.

a) PHP fwrite() Function

The fwrite function is used to write files.

It has the following syntax

```
<?php
fwrite($handle, $string, $length);
?>
```

HERE,

- "fwrite" is the PHP function for writing to files
- "$handle" is the file pointer resource
- "$string" is the data to be written in the file.
- "$length" is optional, can be used to specify the maximum file length.

b) PHP fread() Function

The fread() reads from an open file.

The function will stop at the end of the file or when it reaches the specified length, whichever comes first.

Syntax

```
fread(file, length)
```

Parameter Values

Parameter	Description
File	Required. Specifies the open file to read from
Length	Required. Specifies the maximum number of bytes to read

c) Append a file

The append mode is used to add the text in the file. The code below illustrates the implementation. If the file doesn't exist attempt to create it.

```
<?php

fopen($filename,a);

?>
```

3. Deleting a file

The unlink function is used to delete the file. The code below illustrates the implementation.

```
<?php
unlink($handle);
?>
```

- "" is the PHP function for closing an open file
- "$handle" is the file pointer resource.

4. PHP fclose() Function

The fclose() function is used to close a file in php which is already open

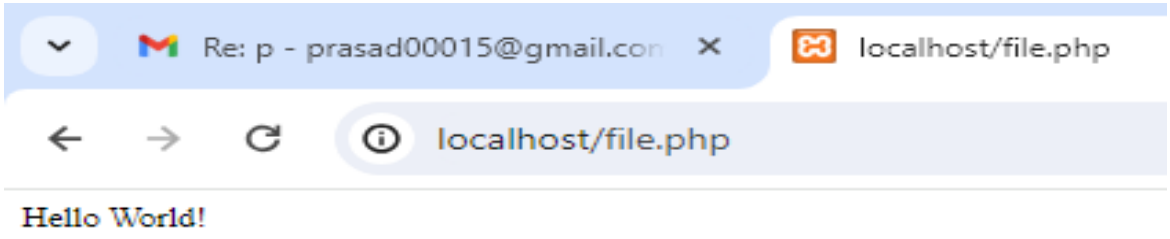
It has the following syntax.

```
<?php
fclose($handle);
?>
```

HERE,

- "fclose" is the PHP function for closing an open file
- "$handle" is the file pointer resource.

```
<?php
if(file_exists("demo.txt")){
   $fp=fopen("demo.txt","r");
   $r=fread($fp,filesize("demo.txt"));
   echo $r;
   fclose($fp);
}
else{
$f=fopen("demo.txt","w");
fwrite($f,"Hello World!");
echo "file created sucessfully";
fclose($f);
}
?>
```



❖ Files using html and css

To save : **formHandling.html**

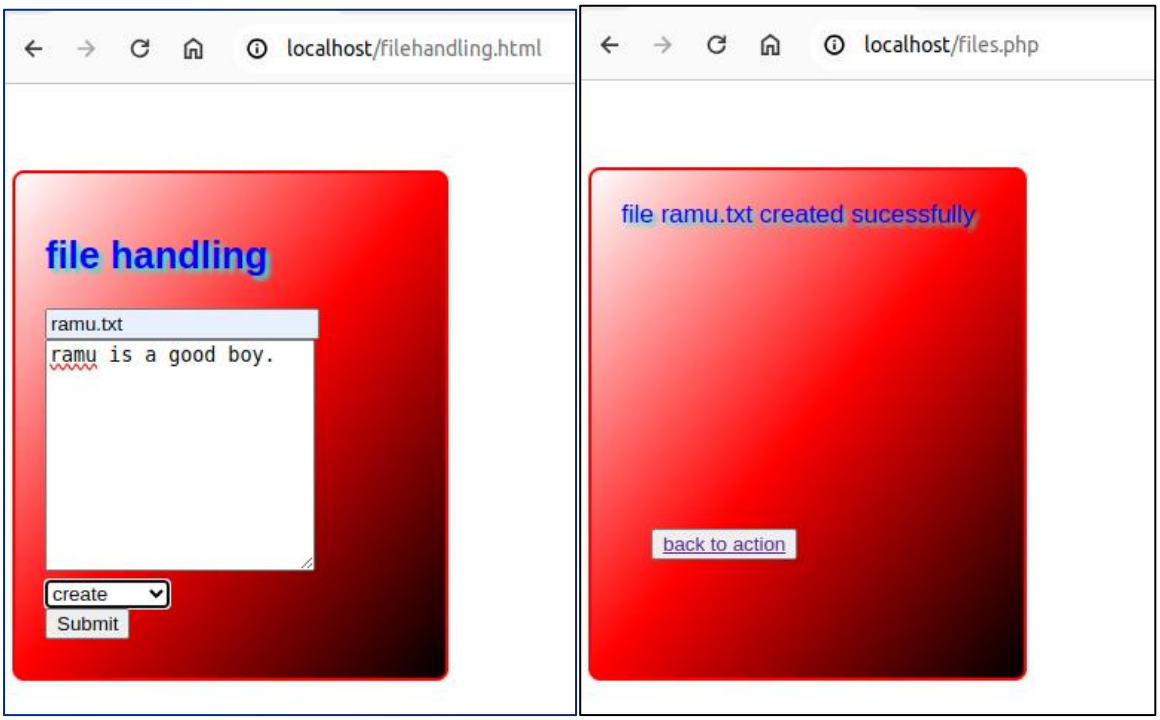
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Document</title>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
<div>
<h2>file handling</h2>
<form action="files.php" method="post">
<input type="text" name="filename" placeholder="filename"><br>
<textarea name="content" placeholder="enter text" id="" cols="20" rows="10"></textarea><br>
<select name="task" id="">
<option>---select---</option>
<option value="create">create</option>
<option value="open">open</option>
<option value="append">append</option>
<option value="delete">delete</option>
</select><br>
<input type="submit" name="submit">
</form>
</div>
```

```
</body>

</html>
```

To save :**Files.php**

```
<?php

if(isset($_POST['submit'])){

   $name=$_POST['filename'];

   $action=$_POST['task'];

   if(strcmp($action,"open")==0){

      if(file_exists($name)){

      $fp=fopen($name,'r');

      $r=fread($fp,filesize($name));

      echo $r;

      fclose($fp);

      }

      else{

      echo 'file not exists';

      }


   }

   elseif(strcmp($action,"create")==0){

      $content=$_POST['content'];

      $fp=fopen($name,'w');

      fwrite($fp,$content);
```

```
    echo "file $name created sucessfully";
  }
  elseif(strcmp($action,"append")==0){
    $content=$_POST['content'];
    $fp=fopen($name,'a');
    fwrite($fp,$content);
    echo "file $name changed sucessfully";
    fclose($fp);
  }
  elseif(strcmp($action,"delete")==0){
    if(file_exists($name)){
    unlink($name);
    echo "file $name deleted sucessfully";
    fclose($fp);
    }
    else{
      echo "file not found";
    }
  }
  else{
    echo "commands not working";
  }
}
?>
```



❖ PHP Directory

Introduction to PHP Directory

PHP directory functions as their name suggests are a set of functions used in retrieving details, modifying them and fetching information on various file system directories and their specific contents. A lot of operations can be performed on the directories like creating, deleting, changing the present working directory, listing files present in the directory and so on. There is no separate installation required for these functions as they come as part of the PHP core. But to enable chroot() function we need to configure –enable-chroot-func option.

Functions of PHP Directory

Let us go through a few of the basic PHP directory functions as below:

1. Create a New Directory

We use the mkdir() function to create a new directory in the PHP programming script.

Syntax:

```
mkdir($dir_path,$mode,$recursive_flag,$context);
```

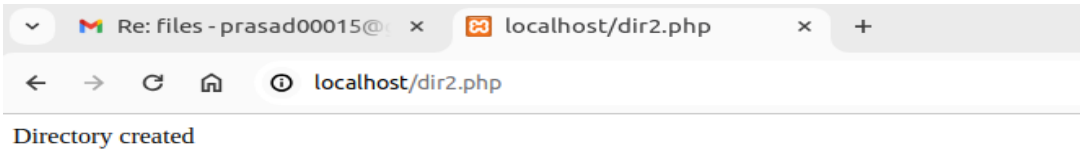
where,

- $**dir_path** is either the relative or the absolute path where the new directory specified will be created.
- $**mode** is the parameter that will take in octal values which determines the level in which the newly created directory will be accessible.
- $**recursive is a flag** type field that has 2 values either true or false which can either allow us to create nested directories or not.
- $**context** is similar to what we have with PHP unlink() like having a stream to specify certain protocols etc. This will also return only a boolean value which will be true if the execution is completed successfully and false otherwise.
-

Example:

```
<?php
mkdir("articles/");
echo("Directory created");
?>
```

Output:



This is a basic example to show the creation of a directory in the path we require. Make sure the path has sufficient permissions else “permission denied” error will be thrown.

2. List the Contents of a Directory

We use opendir() and readdir() for opening the directory link and to read it respectively. Step 1 will be to open the directory and Step 2 will be to read it.

Step 1: To open the directory link, opendir() is the function we use to do this step. It requires two input arguments as specified below.

Syntax:

```
opendir($dir_path,$context);
```

- $**dir_path** is the path of the directory which needs to be opened.
- $**context** is an optional parameter where we can specify if a context stream is present.

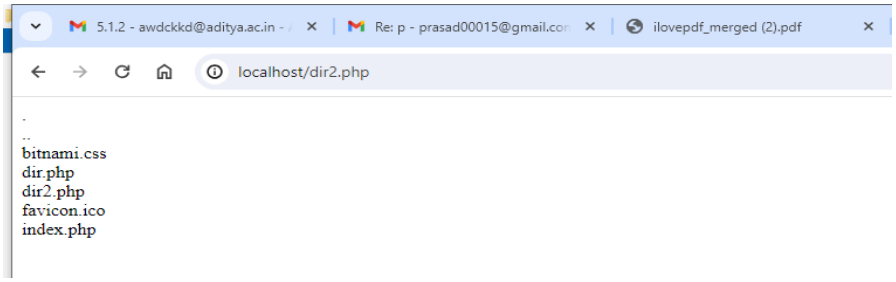
This returns resource data value as its output. This resource ID which it provides is used in our further processing steps else we get an error as resource ID is invalid.

Step 2: To read the contents of the directory, readdir() is the function which is used for this purpose and it needs to be called recursively till the end of the directory is reached by the directory handle.

Example:

```
<?php
$direct = "articles/";
if (is_dir($direct)){
if ($td = opendir($direct)){
while (($file = readdir($td)) !== false){
echo   $file . "<br>";
}
closedir($td);
}
}
?>
```

Output:



In this example first, we are declaring the directory path which needs to be read. We are checking in the if statement if the directory is present and then proceeding to open the contents of the directory and read. The output displays the filenames present inside the directory.

3. To Close a Directory

We use closedir() function in order to close a directory after reading its contents.

Syntax:

```
$dir_handle = opendir($dir_path);
...
...
closedir($dir_handle);
```

In this example, we are first declaring the path of our directory. Then using the if conditional statement we are checking if the path is valid and if yes, then we are opening the directory, reading its variables and then closing it. Thus, any operation can be done between the **opening and closing** of the directory.

4. To Change the Current Directory

We use the function chdir() to change the current working directory in which it is pointing to.

Syntax:

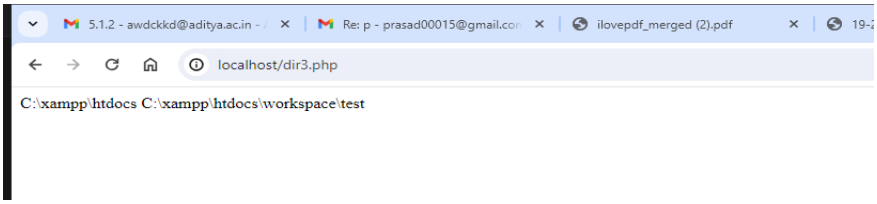
```
chdir(directory)
```

It requires only one parameter that is the directory to which the current working directory should be pointed to. It returns true on success and false if failed to change the directory.

Example:

```
<?php
// Get current directory
echo getcwd()."\n";
// Change directory
chdir("workspace/test");
// Get current directory
echo getcwd();
?>
```

Output:



In this example, we are first printing the present working directory. Then we are changing the same using chdir function to “test” directory and printing the same on the output. Hence make sure the entire path we are giving here exists.

5. Delete Directory

For this purpose, we are using remdir() function which can remove the directory.

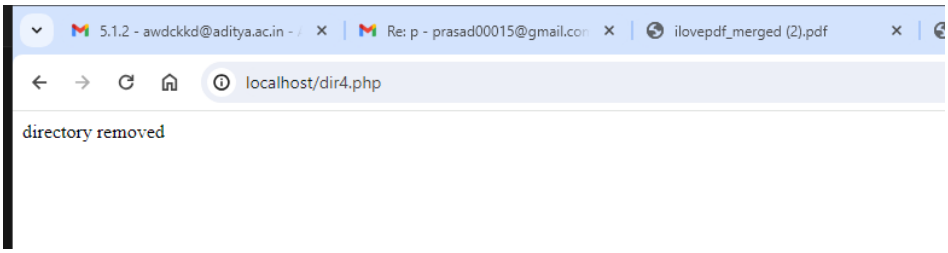
Syntax:

```
remdir(directory)
```

Example:

```
<?php
$dir="articles";
//remove directory
if(rmdir($dir)==true){
   echo "directory removed";
}
else{
   echo "directory not found";
}
```

Output:



❖ Image creation

The imagecreate() function is used to create a new image. It is preferred to use imagecreatetruecolor() to create an image instead of imagecreate(). This is because the image processing occurs on the highest quality image possible which can be created using imagecreatetruecolor().

Syntax

imagecreate($width, $height)

Parameters

width: The width of image
height: The height of image

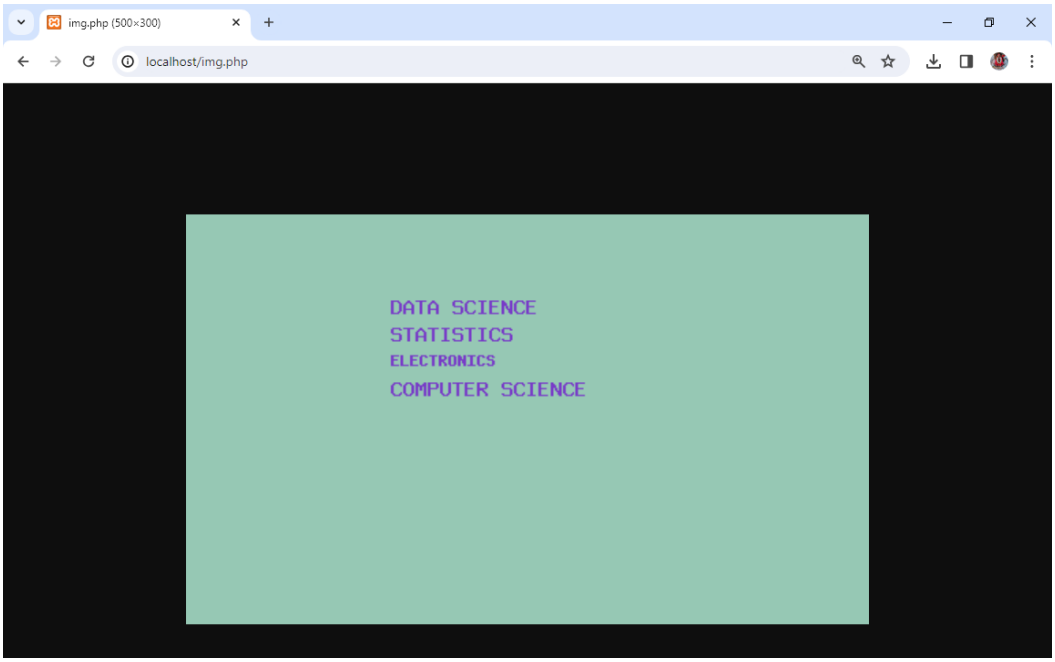
Return

The imagecreate() function returns an image resource identifier on success or FALSE on errors.

Example

The following is an example:

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



❖ PHP | imagepolygon() Function

The **imagepolygon()** function is an inbuilt function in PHP which is used to draw a polygon. This function returns TRUE on success and returns FALSE otherwise.

Syntax:

bool imagepolygon($image, $points, $num_points, $color)

Parameters: This function accepts four parameters as mentioned above and described below:

- **$image:** The imagecreatetruecolor() function is used to create a blank image in a given size.
- **$points:** This parameter is used to hold the consecutive vertices of polygon.
- **$num_points:** This parameter contains total number of vertices in a polygon. It must be greater than 3, because minimum three vertices required to create a polygon.
- **$color:** This variable contains the filled color identifier. A color identifier created with imagecolorallocate() function.

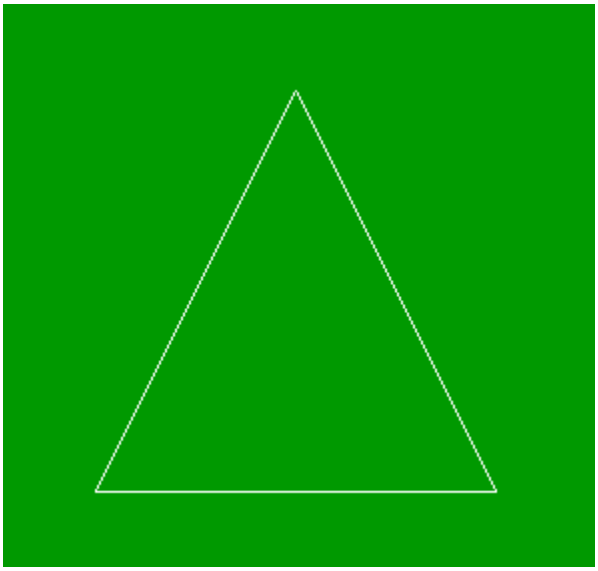
Return Value: This function returns TRUE on success or FALSE on failure.
Below programs illustrate the **imagepolygon()** function in PHP.

Program :

```
<?php

// Set the vertices of polygon
$values = array(
```

```
        150, 50, // Point 1 (x, y)
        50, 250, // Point 2 (x, y)
        250, 250 // Point 3 (x, y)
    );

// Create the size of image or blank image
$image = imagecreatetruecolor(300, 300);

// Set the background color of image
$background_color = imagecolorallocate($image, 0, 153, 0);

// Fill background with above selected color
imagefill($image, 0, 0, $background_color);

// Allocate a color for the polygon
$image_color = imagecolorallocate($image, 255, 255, 255);

// Draw the polygon
imagepolygon($image, $values, 3, $image_color);

// Output the picture to the browser
header('Content-type: image/png');

imagepng($image);
?>
```

Output:



Unit-V

❖ MySQL vs MySQLi

MySQL and MySQLi are PHP database extensions implemented by using the PHP extension framework. PHP database extensions are used to write PHP code for accessing the database.

MySQL extension is deprecated and will not be available in future PHP versions. It is recommended to use the MySQLi extension with PHP 5.5 and above.

MYSQLi	MYSQL
MySQLi extension added in PHP 5.5	MySQL extension added in PHP version 2.0
Extension directory: ext/mysqli.	Extension directory: ext/mysql
The MySQLi supports prepared statements.	The MYSQL does not support prepared statements.
MySQLi supports transactions through API.	Transactions are handled by SQL queries only.
MySQLi provides both object-oriented and procedural interfaces.	MySQL provides procedural interface.
MySQLi extension is with enhanced security and improved debugging.	MySQL extension lags in security and other speci features, comparatively.
MySQL provides a procedural interface.	MySQLi provides both object oriented and procedural interface.
MySQLi supports store procedure.	MySQL extension does not support stored procedure.
ysqli_connect($host_name,$user_name,$passwd,$dbname)	mysql_connect($host_name,$user_name,$passw followed by mysql_select_db($dbname)

❖ Connect PHP to MySQL

MySQL

MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP.

Structured Query Language (SQL). The data in a MySQL database are stored in tables that consist of columns and rows.

MySQL is a database system that runs on a server. MySQL is ideal for both small and large applications. MySQL is a very fast, reliable, and easy-to-use database system. It uses standard SQL. MySQL compiles on a number of platforms.

How we can connect PHP to MySQL?
PHP 5 and later can work with a MySQL database using:

- MySQLi extension (the 'i' is abbreviation for improved)
- PDO (PHP Data Objects)
- Both are object-oriented, but MySQLi also offers a procedural API.

Connection to MySQL using MySQLi
PHP provides mysql_connect() function to open a database connection.

This function takes a single parameter, which is a connection returned by the mysql_connect() function.

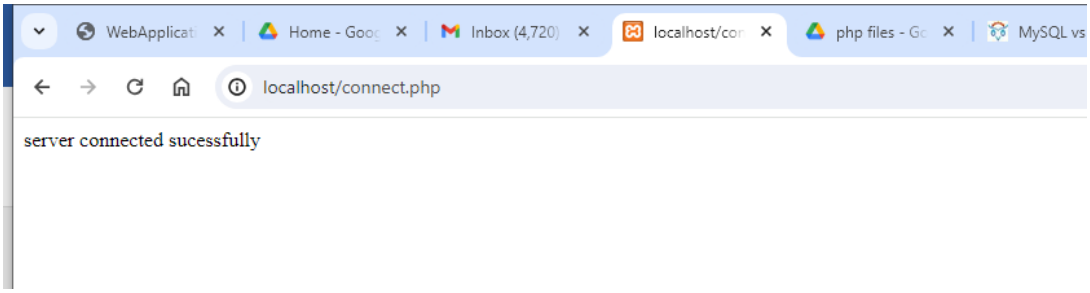
You can disconnect from the MySQL database anytime using another PHP function mysql_close().

There is also a procedural approach of MySQLi to establish a connection to MySQL database from a PHP script.

It can be done in two ways:

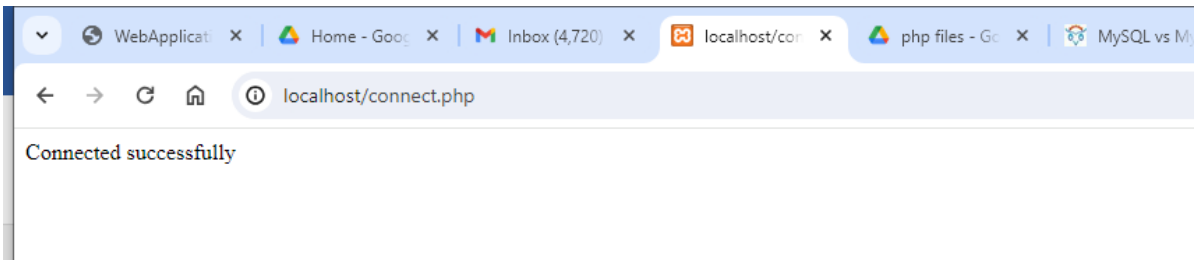
MySQLi Object-Oriented

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

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
else
{
  echo "server connected sucessfully";
}
```



MySQLi Procedural

```php
<?php

$servername = "localhost";
$username = "root";
$password = "root";

// Connection
$conn = mysqli_connect($servername, $username, $password);

// Check if connection is
// Successful or not
if (!$conn) {
  die("Connection failed: ". mysqli_connect_error());
}
echo "Connected successfully";
?>
```



❖ Creating Database

Database

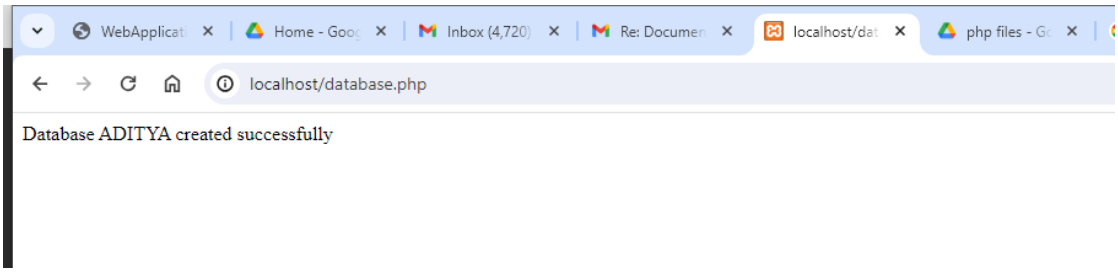
Database is a collection of inter-related data which helps in efficient retrieval, insertion

and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.
We know that in MySQL to create a database we need to execute a query.
The basic steps to create MySQL database using PHP are:

- Establish a connection to MySQL server from your PHP script as described.
- If the connection is successful, write a SQL query to create a database and store it in a string variable.
- Execute the query.

Using MySQLi Object-oriented procedure: If the MySQL connection is established using Object-oriented procedure then we can use the query() function of mysqli class to execute our query as described in the below.

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

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE ADITYA";
if ($conn->query($sql) === TRUE) {
  echo "Database ADITYA created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```



Note: Specify the three arguments servername, username and password to the mysqli object whenever creating a database.

❖ PHP MySQL Create Table

A database table has its own unique name and consists of columns and rows.

Create a MySQL Table Using MySQLi

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "student", with five columns: "sno", "studentname", "email", "gender" and "age":

```
CREATE table student(
   Sno INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   studentname VARCHAR(30) NOT NULL,
   email VARCHAR(30) NOT NULL,
   gender VARCHAR(50),
   age INT(2)

   )
```

Notes on the table above:

The **data type** specifies what type of data the column can hold. For a complete reference of all the available data types, go to our Data Types reference.

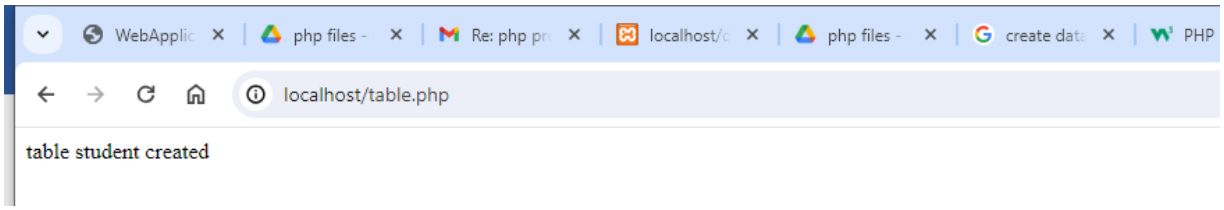
After the data type, you can specify other **optional attributes** for each column:

- **NOT NULL** - Each row must contain a value for that column, null values are not allowed
- **DEFAULT value** - Set a default value that is added when no other value is passed
- **UNSIGNED** - Used for number types, limits the stored data to positive numbers and zero
- **AUTO INCREMENT** - MySQL automatically increases the value of the field by 1 each time a new record is added
- **PRIMARY KEY** - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "sno" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

```
<?php
$servername="localhost";
$username="root";
$password="root";
$dbname="ADITYA";
$conn=new mysqli($servername,$username,$password,$dbname);
if($conn->connect_error){
   die("error".$conn->connect_error);
}
$sql="CREATE table student(
   Sno INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   studentname VARCHAR(30) NOT NULL,
   email VARCHAR(30) NOT NULL,
   gender VARCHAR(50),
   age INT(2)
   )";
if($conn->query($sql)==true){
   echo "table student created";
}
else{
   echo "error".$conn->error;
}
```



❖ PHP | Inserting into MySQL database

INSERT INTO statement is used to insert new rows in a database table. Let's see the syntax how to insert into table, considering database already exists.

SYNTAX :

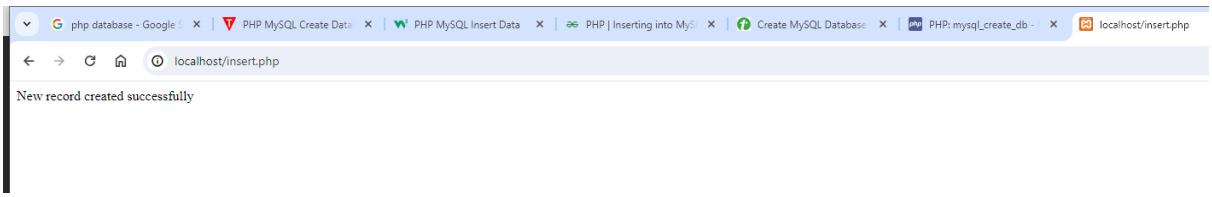
```
INSERT INTO TABLE_NAME (column1, column2, column3, ... columnN)
VALUES (value1, value2, value3, ...valueN);
```

Here, column1, column2, column3, ...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

So to create a SQL query using the INSERT INTO statement with appropriate values, here's an example, which will insert a new row to the **student** table by specifying values for the studentname, email, gender and age fields.

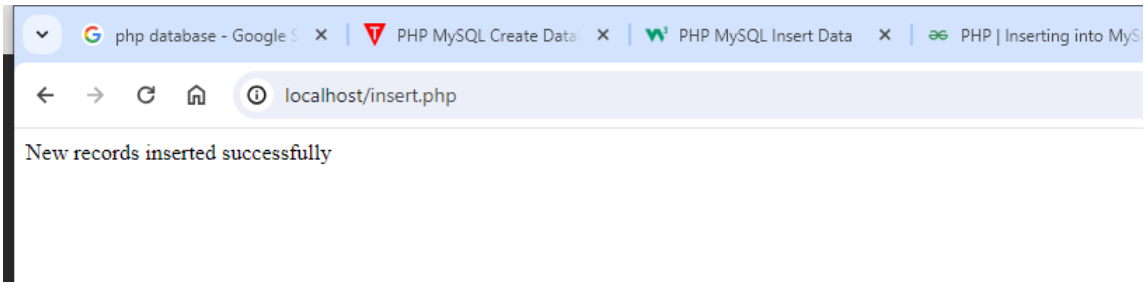
```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "ADITYA";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO student (studentname, email, age, gender)
VALUES ('Ramu','ramu@gmail.com',25,'male')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```



Inserting Multiple Rows into a Table

One can also insert multiple rows into a table with a single insert query at once. To do this, include multiple lists of column values within the INSERT INTO statement, where column values for each row must be enclosed within parentheses and separated by a comma.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "ADITYA";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO student (studentname, email, age, gender)VALUES
    ('ravi','ravi@gmail.com',26,'male'),
    ('ajay','ajay@gmail.com',25,'male')";

if ($conn->query($sql) === TRUE) {
  echo "New records inserted successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```



❖ Select Data From MySQL Database

Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function mysql_query. You have several options to fetch data from MySQL.

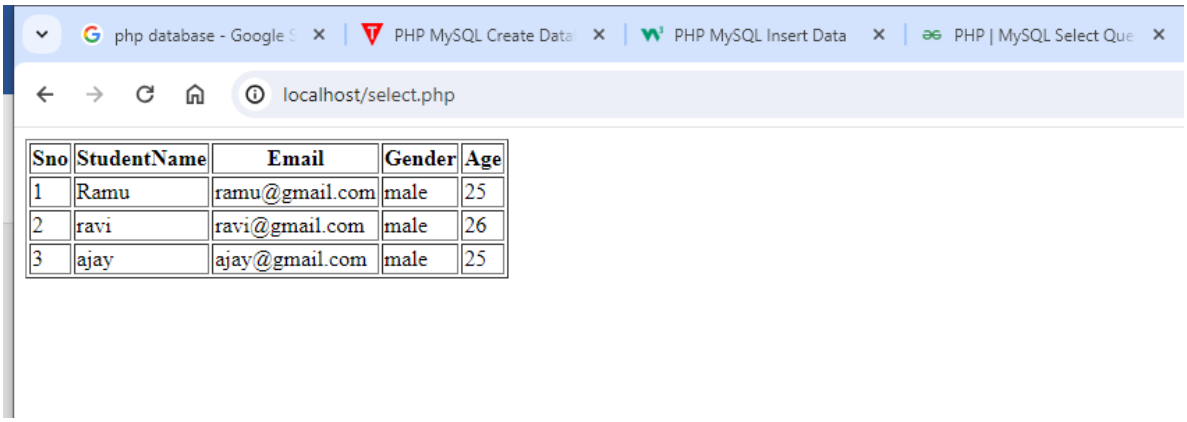
The most frequently used option is to use function **mysql_fetch_assoc()** which also returns the row as an associative array.

Example

Try out following example to display all the records from **student** table using mysql_fetch_assoc() function.

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "ADITYA";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM student";
$result = $conn->query($sql);
echo "<table border=1>

  <tr>


  <th>Sno</th>

  <th>StudentName</th>

  <th>Email</th>
  <th>Gender</th>
  <th>Age</th>

  </tr>";
```

```
if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "<tr>";

    echo "<td>" . $row['Sno'] . "</td>";

    echo "<td>" . $row['studentname'] . "</td>";

    echo "<td>" . $row['email'] . "</td>";
    echo "<td>" . $row['gender'] . "</td>";
    echo "<td>" . $row['age'] . "</td>";

    echo "</tr>"; }
} else {
  echo "0 results";
}
$conn->close();
?>
```



Sno	StudentName	Email	Gender	Age
1	Ramu	ramu@gmail.com	male	25
2	ravi	ravi@gmail.com	male	26
3	ajay	ajay@gmail.com	male	25

You can also use the constant **MYSQL_NUM**, as the second argument to mysql_fetch_array(). This will cause the function to return an array with

❖ Deleting Database Table Data

Just as you insert records into tables, you can delete records from a table using the SQL DELETE statement. It is typically used in conjugation with the `WHERE` clause to delete only those records that matches specific criteria or condition.

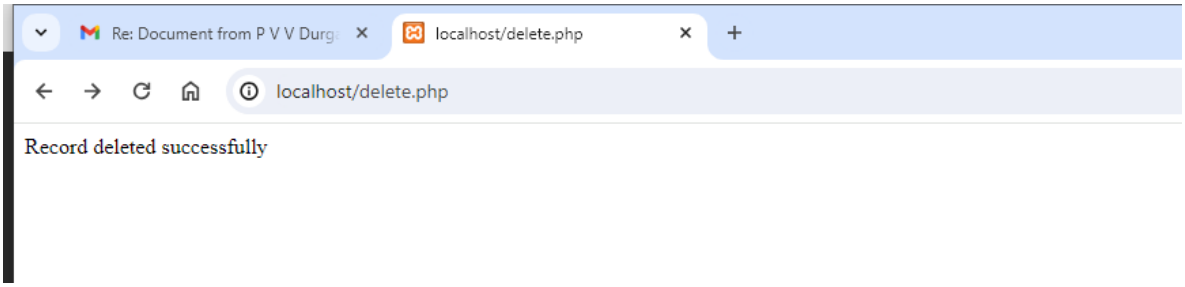
The basic syntax of the `DELETE` statement can be given with:

```
DELETE FROM table_name WHERE column_name=some_value
```

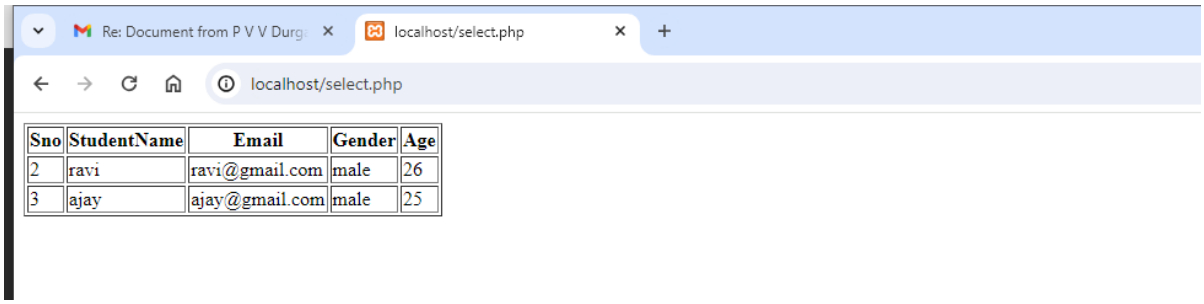
Let's make a SQL query using the `DELETE` statement and `WHERE` clause, after that we will execute this query through passing it to the PHP `mysqli_query()` function to delete the tables records. Consider the following **student** table inside the **ADITYA** database:

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.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "ADITYA";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM student WHERE Sno =1";

if ($conn->query($sql) === TRUE) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```



Delete after select records in **student** table



Sno	StudentName	Email	Gender	Age
2	ravi	ravi@gmail.com	male	26
3	ajay	ajay@gmail.com	male	25

❖ Student Data project

Forms.html

```
<html>
  <head><title>FormHandling</title>

    <link rel="stylesheet" href="mystyle.css">
    </head>
  <body>
    <div class="container">
      <h3 class="text">Student Registration</h3>
    <form method="POST"action="insertmystudent.php">
      <input type="text"name="name"placeholder="Name"><br><br>
      <input type="email"name="email"placeholder="E-Mail"><br><br>
      <input type="number"name="age"placeholder="Age"><br><br>
      <input type="radio"name="gender"value="Male">Male
      <input type="radio"name="gender"value="Female">Female<br>

      <input type="submit"name="submit"value="Submit"><br><br>
      <button><a href="selectmystudents.php">results</a> </button>
      </form>

      </div>



  </body>
</html>
```

Mystyle.css

```
 body{
 background-color: #666;
}
.container {
            position:absolute;
            top: 50%;
            left: 50%;
          transform: translate(-50%,+50%);

            top:30px;

    box-sizing: border-box;
    padding: 15px;
    border-radius: 0.5rem;
    background-image:linear-gradient(to right top,hsl(0, 16%, 16%),hsl(261, 62%, 51%));
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    color:rgb(245, 54, 20);

  }
  .text{
     font-family:'Times New Roman', Times, serif;
     color:rgb(221, 195, 195);
     text-align:center;

  }

  table {
     margin:0px;

}

th {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 3em;
   background: #666;
   color: #FFF;
   padding: 2px 6px;

}

td {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 2em;
     color:rgb(182, 235, 209);
     text-shadow: 2px 3px 2px rgb(27, 27, 26);
}
```

```
tr {
  background-image:linear-gradient(to right top,rgb(250, 250, 250),rgb(12,0,150));
}
```

Connects.php

```
<?php
$servername="localhost";
$username="root";
$password="root";
$conn=new mysqli($servername,$username,$password);
if($conn->connect_error){
   die("error".$conn->connect_error);
}
$sql="CREATE DATABASE php";
if($conn->query($sql)==true){
   echo "database PHP created";
}
else{
   echo "error".$conn->error;
}

?>
```

Mystudents.php

```
<?php
$servername="localhost";
$username="root";
$password="root";
$dbname="php";
$conn=new mysqli($servername,$username,$password,$dbname);
if($conn->connect_error){
   die("error".$conn->connect_error);
}
$sql="CREATE table studentphp(Sno INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
studentname VARCHAR(30) NOT NULL,
email VARCHAR(30) NOT NULL,
gender VARCHAR(50),
age INT(2))";
if($conn->query($sql)==true){
   echo "table student php created";
}
else{
   echo "error".$conn->error;
}
```

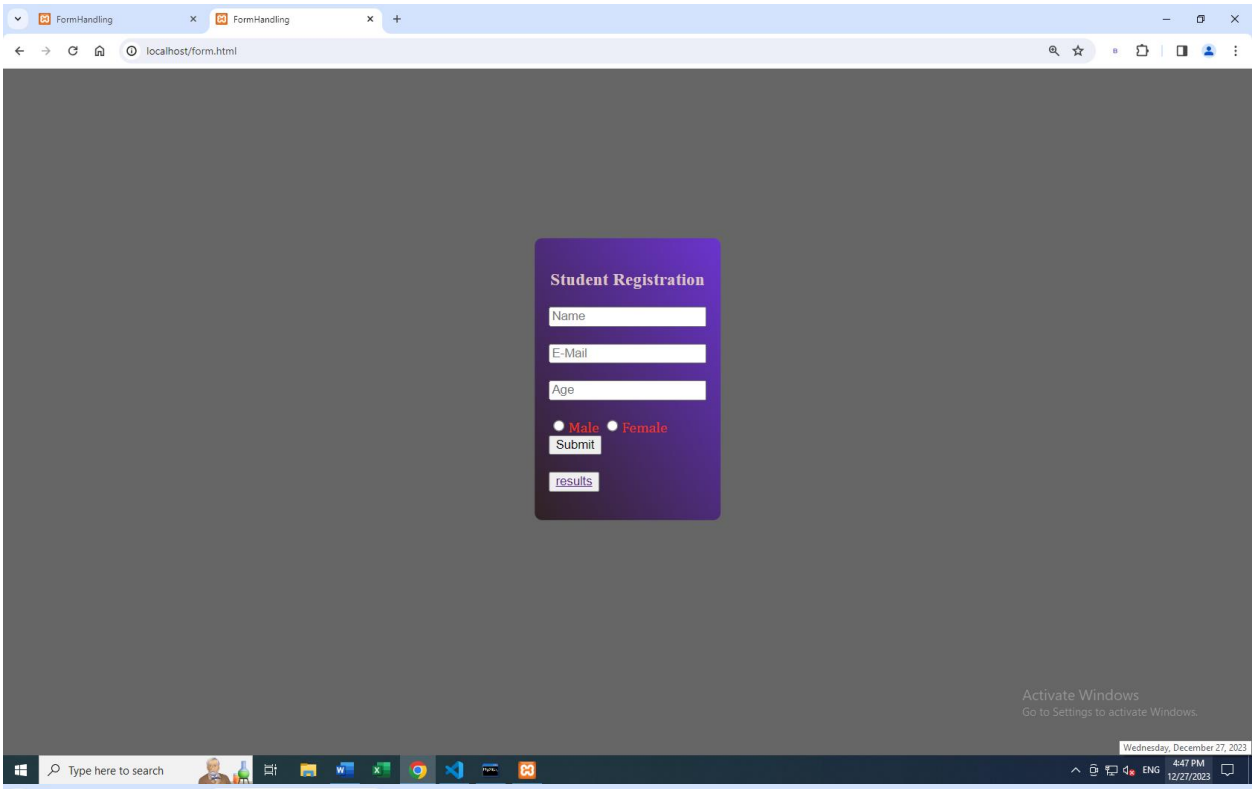
Insertmystudents.php

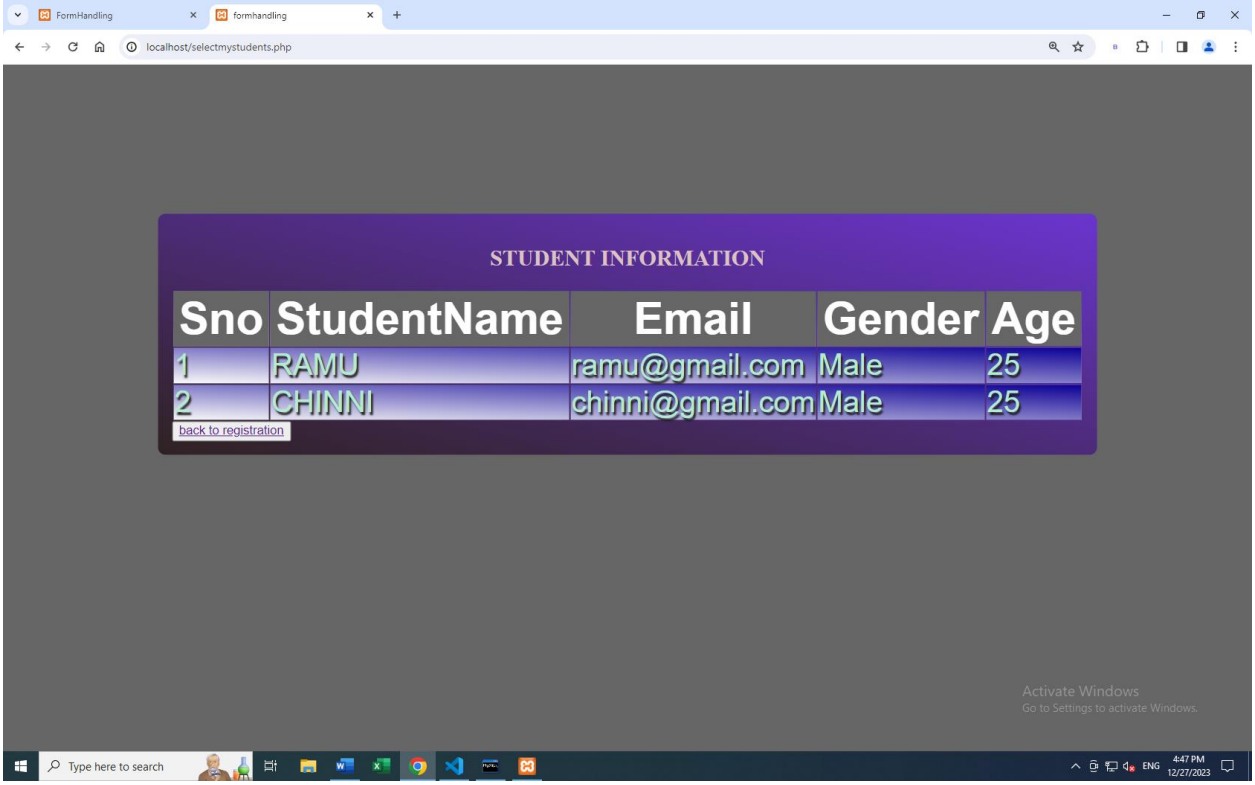
```
<html>
   <head>
      <title>insert</title>
      <link rel="stylesheet" href="mystyle.css">
</head>
<body>
   <div class="container">
<?php
$servername="localhost";
$username="root";
$password="root";
$dbname="php";
$conn=new mysqli($servername,$username,$password,$dbname);
if($conn->connect_error){
   die("error".$conn->connect_error);
}
$name=$_POST['name'];
$email=$_POST['email'];
$age=$_POST['age'];
$gender=$_POST['gender'];
if(empty($name)) {
   echo "enter student name";
}
else if(empty($email)) {
   echo "enter email id";
}
else if(empty($age)) {
   echo "enter age";
}
else if(empty($gender)) {
   echo "select gender";
}
else
{
$sql="insert into studentphp(studentname,email,gender,age)
values('$name','$email','$gender','$age')";
if($conn->query($sql)==true)
{
   echo "record inserted";
}
else
echo"please enter record";
}
?>
<br>
<br>
<a href="form.html">Go Back</a><br>
<a href="selectmystudents.php">result</a>
</div>
</body>
```

Selectstudents.php

```
<html>
   <head>
      <title>formhandling</title>
      <link rel="stylesheet" href="mystyle.css">
   <style>
     <style type=”text/css”>


</style>

   </style>
   </head>
<body>
<div class="container">
   <h2 class="text"> STUDENT INFORMATION</h2>
  <?PHP
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "php";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM studentphp";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  echo "<table>

  <tr>


  <th>Sno</th>

  <th>StudentName</th>

  <th>Email</th>
  <th>Gender</th>
  <th>Age</th>
```

```
 </tr>";



 while($row = $result->fetch_assoc()) {

  echo "<tr>";


 echo "<td>" . $row['Sno'] . "</td>";

 echo "<td>" . $row['studentname'] . "</td>";

 echo "<td>" . $row['email'] . "</td>";
 echo "<td>" . $row['gender'] . "</td>";
 echo "<td>" . $row['age'] . "</td>";

 echo "</tr>";

 }
}

echo "</table>";

$conn->close();
?>
<button><a href="form.html">back to registration</a></button>
</div>
</body>
</html>
```


FormHandling
FormHandling
localhost/form.html
Student Registration
Name
E-Mail
Age
Male
Female
Submit
results
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
4:47 PM
12/27/2023


FormHandling
formhandling
localhost/selectmystudents.php
STUDENT INFORMATION
Sno
StudentName
Email
Gender
Age
1
RAMU
ramu@gmail.com
Male
25
2
CHINNI
chinni@gmail.com
Male
25
back to registration
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
4:47 PM
12/27/2023