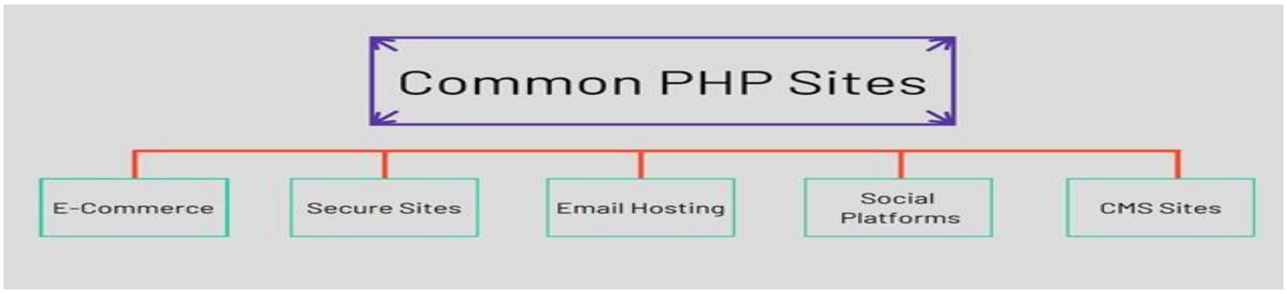
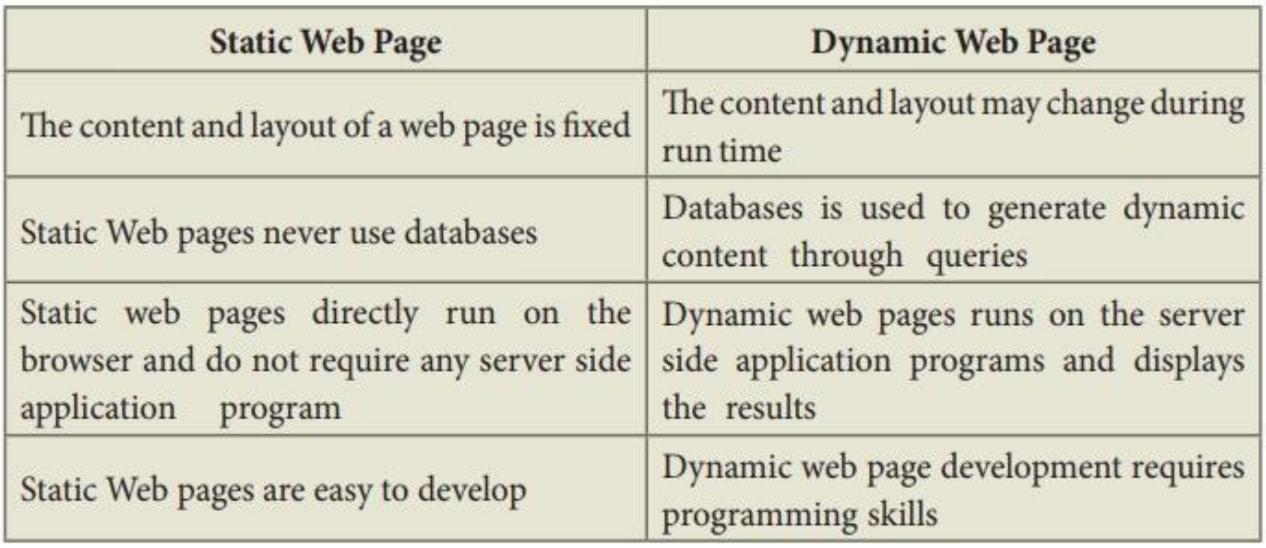
Introduction to PHP and MYSQL

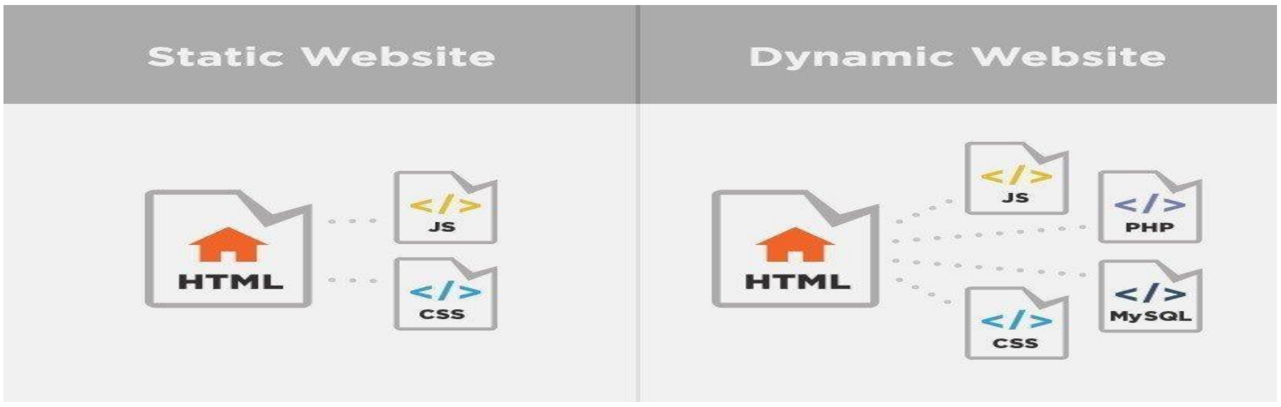
What is PHP:

- PHP stands for Hypertext Pre-processor.
- PHP is a server-side scripting language that is used to develop Static websites or Dynamic websites or Web applications.
- PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995.
- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP scripts can only be interpreted on a server that has PHP installed.
- PHP scripts are executed on the server and the result is sent to the browser as plain HTML.
- PHP can be embedded within a normal HTML web pages. That means inside your HTML documents you'll have PHP statements
- PHP can be integrated with the number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.





Static Web Page	Dynamic Web Page
The content and layout of a web page is fixed	The content and layout may change during run time
Static Web pages never use databases	Databases is used to generate dynamic content through queries
Static web pages directly run on the browser and do not require any server side application program	Dynamic web pages runs on the server side application programs and displays the results
Static Web pages are easy to develop	Dynamic web page development requires programming skills

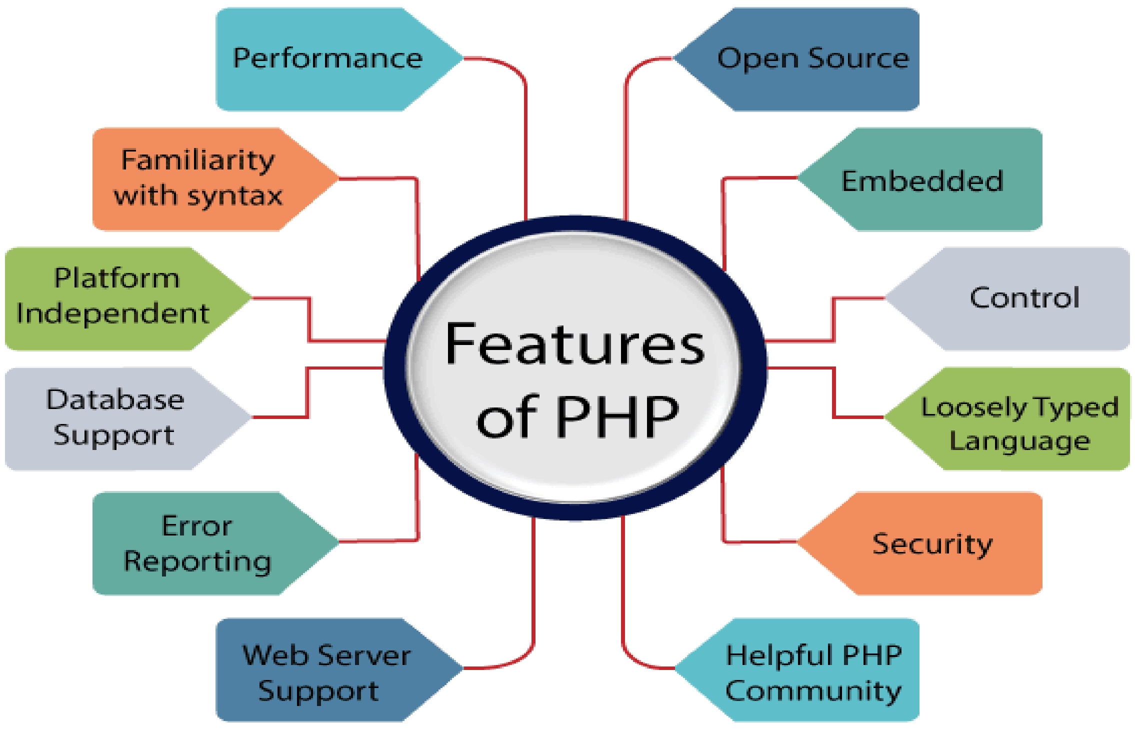


What You Can Do with PHP

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

PHP Features

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



Performance:

- PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP.
- PHP uses its own memory, so the server workload and loading time is automatically 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, etc.

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. Learning PHP from the communities is one of the significant benefits.

Disadvantages of PHP

Despite the advantages of the PHP Hypertext Pre processor, the scripting language also has some disadvantages. Some of the disadvantages are explained below.

- **Security** : Since it is open sourced, so all people can see the source code, if there are bugs in the source code, it can be used by people to explore the weakness of PHP
- **Not suitable for large applications**: PHP is not highly modular, huge applications created out of the programming language will be difficult to maintain.
- **Weak type:** Implicit conversion may surprise unwary programmers and lead to unexpected bugs. For example, the strings "1000" and "1e3" compare equal because they are implicitly cast to floating point numbers.

MySQL:

- MySQL is a fast, easy to use relational database.
- It is currently the most popular open-source database.
- It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.
- MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company. It is written in C and C++.

MySQL is becoming so popular because of these following reasons:

- MySQL is an open-source database so you don't have to pay a single penny to use it.
- MySQL is a very powerful program so it can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open source database and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MySQL Features

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system.
- **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- **Free to download:** MySQL is free to use and you can download it from MySQL official website.
- **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **Compatible on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.
- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Disadvantages / Drawback of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

PHP Syntax:

- PHP code is start with **<?php** and ends with **?>**
- Every PHP statements end with a semicolon **(;)**.
- PHP code save with .php extension.
- a PHP file contains HTML tags and some PHP scripting code.
- You can place PHP code any where in your document.

PHP Syntax

```
<?php
// your code here
?>
```

Ex:

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

Embedding PHP within HTML:

PHP files are plain text files with .php extension. Inside a html file you can place php script.

Example: first.php

```
<!DOCTYPE>
<html>
<head>
<title>my first php program</title>
</head>
<body>
<?php
 echo "welcome to computer cluster";
 ?>
 </body>
 </html
```

Execution:

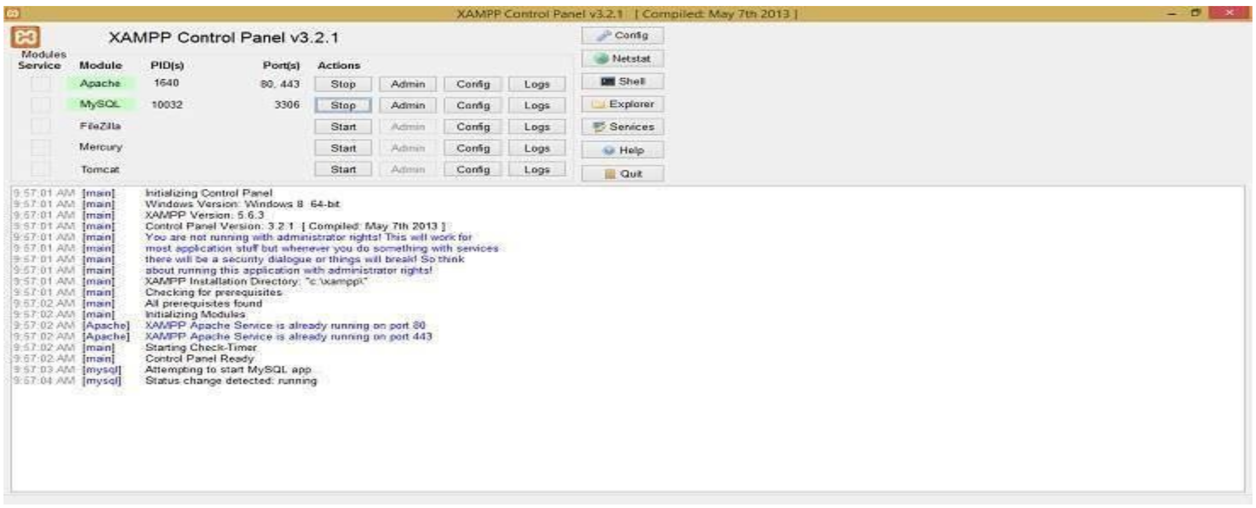
Step1:

Open Mycomputer →C drive →xampp →htdocs →create your folder →save your fitst.php file

Ex: C:\xampp\htdocs\mahesh\first.php

Step2:

double click on "**XAAMP CONTROL PANEL**" on desktop and START "**Apache**"

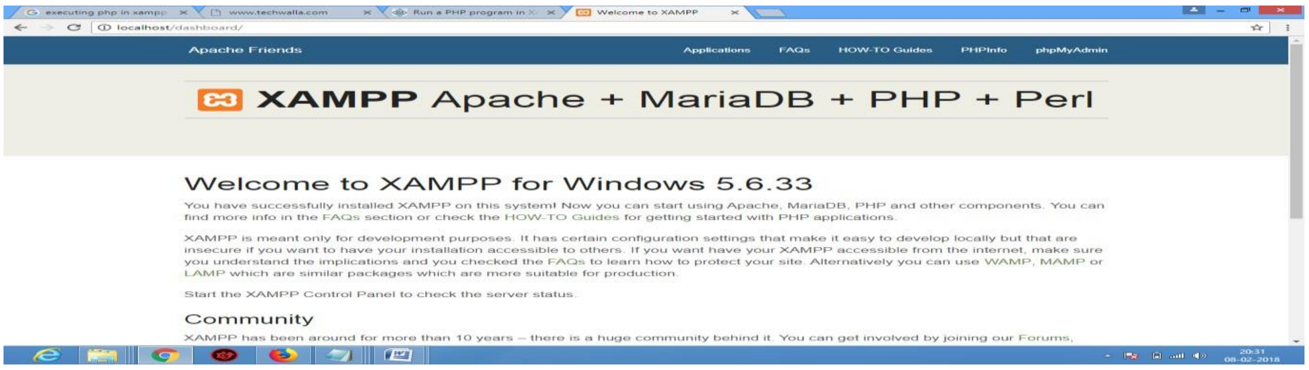

XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]
XAMPP Control Panel v3.2.1
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	1640	80, 443	Stop	Admin	Config	Logs
	MySQL	10032	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
9:57:01 AM  [main]     Initializing Control Panel
9:57:01 AM  [main]     Windows Version: Windows 8  64-bit
9:57:01 AM  [main]     XAMPP Version: 5.6.3
9:57:01 AM  [main]     Control Panel Version: 3.2.1  [ Compiled: May 7th 2013 ]
9:57:01 AM  [main]     You are not running with administrator rights! This will work for
9:57:01 AM  [main]     most application stuff but whenever you do something with services
9:57:01 AM  [main]     there will be a security dialogue or things will break! So think
9:57:01 AM  [main]     about running this application with administrator rights!
9:57:01 AM  [main]     XAMPP Installation Directory: "c:\xampp\"
9:57:01 AM  [main]     Checking for prerequisites
9:57:02 AM  [main]     All prerequisites found
9:57:02 AM  [main]     Initializing Modules
9:57:02 AM  [Apache]   XAMPP Apache Service is already running on port 80
9:57:02 AM  [Apache]   XAMPP Apache Service is already running on port 443
9:57:02 AM  [main]     Starting Check-Timer
9:57:02 AM  [main]     Control Panel Ready
9:57:03 AM  [mysql]    Attempting to start MySQL app
9:57:04 AM  [mysql]    Status change detected: running
```

Step3:

Type **localhost** on your browser and press enter:
It will show the following:


localhost/dashboard/

Apache Friends | Applications | FAQs | HOW-TO Guides | PHPInfo | phpMyAdmin

XAMPP Apache + MariaDB + PHP + Perl

Welcome to XAMPP for Windows 5.6.33

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.

Community

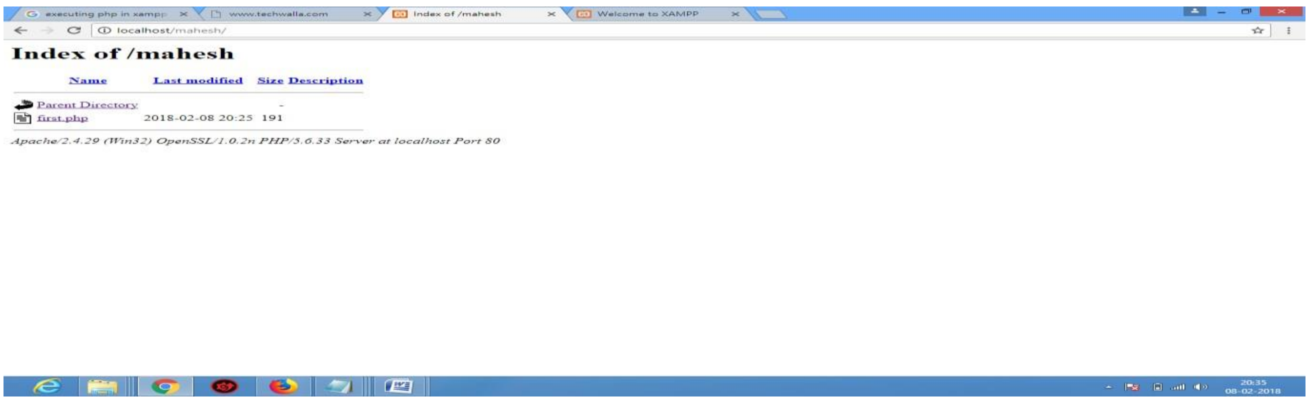
XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums,

Step5:

Now type the following on browser:

<u>localhost/your</u> **folder name/**

EX: localhost/Mahesh

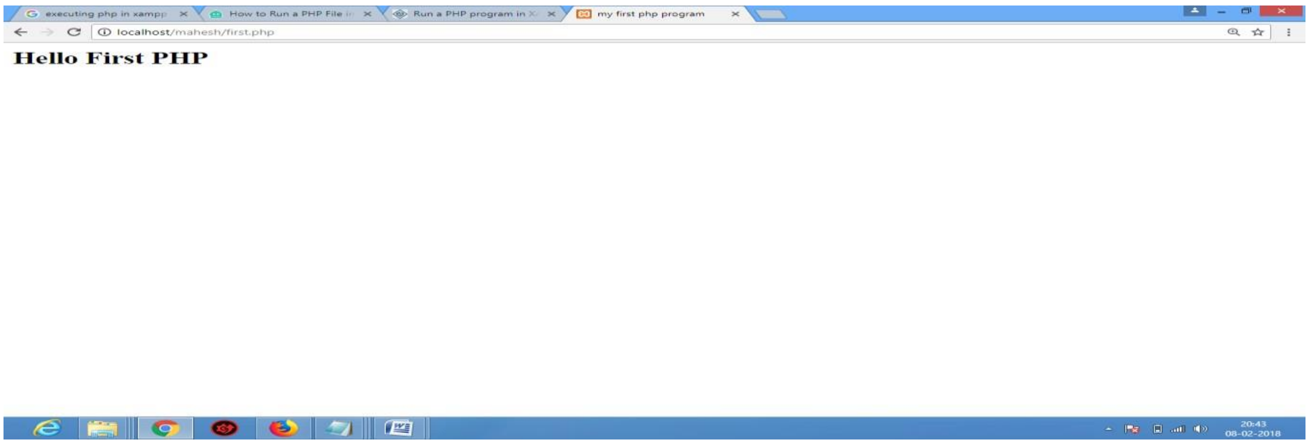

localhost/mahesh/

Index of /mahesh

```
      Name            Last modified      Size  Description
  Parent Directory                         -
  first.php           2018-02-08 20:25   191
Apache/2.4.29 (Win32) OpenSSL/1.0.2n PHP/5.6.33 Server at localhost Port 80
```

Step6

Click on "**first.php**" and it will show the following:


localhost/mahesh/first.php

Hello First PHP

PHP Comments:

- Comments in any programming language is used to describe code.
- Comments are usually written within the block of PHP code to explain the functionality of the code.
- It will help you and others in the future to understand what you were trying to do with the PHP code. Comments are not displayed in the output, they are ignored by the PHP engine.

Types:

- single line Comments # or //
- multi line comments /* .... */

PHP echo & Print:

echo Statement:

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

Syntax: echo (string $arg1 [, string $...]);

Ex:
```
<?php
echo "welcome to computer cluster";
?>
```

PHP print Statement

- print is a statement, used as an alternative to echo 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 the echo statement.

Syntax: print(string $arg)

ex:
```
<?php
 print "welcome to computer cluster";
?>
```

Ex:<?php
```
   $fname  = "mahesh";
   $lname    = "babu";
  print "My name is: $fname”;
  print “$lname”;
?>
```

Ex:
```
<?php
  $fname  = "mahesh";
  $lname    = "babu";
 print"My name is: ".$fname,$lname;
?>
```

PHP Case Sensitivity

- In PHP, keywords (e.g., echo, if, else, while), functions, user-defined functions, classes are not case-sensitive.
- However, all variable names are case-sensitive.
- In the below example, you can see that all three echo statements are equal and valid:

Ex:
```
<!DOCTYPE html>
    <html>
      <body>
        <?php
     echo "Hello world using echo <br>";
    ECHO "Hello world using ECHO <br>";
    EcHo "Hello world using EcHo <br>";
     ?>
     </body>
    </html>
```

Ex:
```
<!DOCTYPE html>
<html>
<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
</body>
</html>
```

Embedding PHP within HTML

- we can embed php file anywhere in our html program.
- Inside a PHP file you can write HTML and inside html program you can write php code.

Ex:

```
<!DOCTYPE html>

<html>
<head>
 <title>A Simple PHP File</title>
</head>
<body>
 <h1>
<?php
echo "Hello, world!"; ?>
</h1>
</body>
</html>
```

Q) The Building blocks of PHP

The following are the Buliding blocks of php

- Variables
- Data Types
- Constants
- Operators

PHP Variables

- Variables are used to store data, like string of text, numbers, etc.
- A variable is a temporary storage that is used to store data temporarily.
- In PHP, a variable is declared using $ sign followed by variable name.

Syntax

```
$variablename=value;
```

Ex:

```
$str="Hello world!";
$a=5;
$b=10.5;
```

PHP Variable Rules:

These are the following rules for naming a PHP variable:

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

Ex: variable1.php

```
<?php
$str="hello string";
$x=200;
```

```
    $y=44.6;
    echo "string is: $str <br/>";
    echo "integer is: $x <br/>";
    echo "float is: $y <br/>";
    ?>
```

PHP Variable: Sum of two variables

File: variable2.php

```
<?php
    $x=5;
    $y=6;
   $z=$x+$y;
    echo $z;
  ?>
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is defined as its range in the program under which it can be accessed.
- PHP has three different variable scopes:
  - local
  - global
  - static

Local variable

- The variables that are declared within a function are called local variables for that function.
- These local variables have their scope only in that particular function in which they are declared.
- This means that these variables cannot be accessed outside the function, as they have local scope.

ex:

```
<?php
 function localvar()
  {
   $num = 45;  //local variable
echo "Local variable declared inside the function is: ". $num;
  }
 localvar();
?>
```

Global variable

- The global variables are the variables that are declared outside the function.
- These variables can be accessed anywhere in the program.
- To access the global variable within a function, use the GLOBAL keyword before the variable.

- However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

Ex:
```
<?php
 $name = "Lakshmi";       //Global Variable
 function globalvar()
 {
 global $name;
  echo "Variable inside the function: ". $name;
 echo "</br>";
  }
 globalvar();
 echo "Variable outside the function: ". $name;
?>
```

Static variable

- when a function is completed/executed, all of its variables are deleted.
- once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution.
- We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.

Ex:
```
  <?php
   function staticvar()
   {
      static $num1 = 3;        //static variable
      $num2 = 6;                          //Non-static variable
                               //increment in non-static variable
      $num1++;
                                //increment in static variable
      $num2++;

      echo "Static: " .$num1 ."</br>";
      echo "Non-static: " .$num2 ."</br>";
   }

//first function call
   staticvar();
   //second function call
   staticvar();
?>
```

<u>PHP $ and $$ Variables:</u>

- The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.
- The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

Ex:

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br>;
echo $$x."<br>";
echo $abc;
?>
```

<u>PHP Data Types:</u>

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)
3. Special Types

Scalar Types: It holds only single value. There are 4 scalar data types in PHP**.**

1. <u>boolean</u>
2. <u>integer</u>
3. <u>float</u>
4. <u>string</u>

Compound Types

It can hold multiple values. There are 2 compound data types in PHP.

1. <u>array</u>
2. <u>object</u>

Special Types

There are 2 special data types in PHP.

1. <u>resource</u>
2. <u>NULL</u>

Boolean

- Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**.

- It is often used with conditional statements.
- If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
  if (TRUE)
     echo "This condition is TRUE.";
  if (FALSE)
     echo "This condition is FALSE.";
?>
```

<u>Integer</u>

- Integer means numeric data with a negative or positive sign.
- It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
- The range of an integer must be lie between - 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

Example:

```
<?php
  $a = 115;
  $b = -109;
   echo "The positive number: ".$a;
  echo "The negative number:".$b;
 ?>
```

<u>Float</u>

- A floating-point number is a number with a decimal point.
- Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

Ex:

```
<?php

 $n1 = 19.34;

$n2 = 54.472;

$sum = $n1 + $n2;

echo "Addition of floating numbers: " .$sum;

?>
```

PHP String

- A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.
- String values must be enclosed either within single quotes or in double quotes.

Ex:

```
<?php
  $college= "shri gnanambica degree college";
 echo "Hello $college";
>?
```

Array

- An array is a compound data type. It can store multiple values of same data type in a single variable.

Ex:

```
<?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>";
?>
```

object

- Objects are the instances of user-defined classes that can store both values and functions.
- They must be explicitly declared.

Ex: < ?php

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

PHP Resource

Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources. For example - a database call. It is an external resource.

PHP Null

Null is a special data type that has only one value: NULL.

The special type of data type NULL defined a variable with no value.

Ex:
```
<?php
    $nl = NULL;
    echo $nl;
   ?>
```

<u>Constants:</u>

- A constant is a name or an identifier for a fixed value.
- Constant are like variables, except that once they are defined, they cannot be undefined or changed
- Constants are very useful for storing data that doesn't change during the execution of the script.

PHP constants can be defined by 2 ways:

- Using define() function
- Using const keyword

PHP constants follow the same PHP variable rules. For example, it can be started with letter or underscore only.

define() :

define(name, value, case-insensitive)

name: specifies the constant name
value: specifies the constant value
case-insensitive: Default value is false. It means it is case sensitive by default.

Ex1:
```
<?php

define("MESSAGE","welcome to PHP");

echo MESSAGE;
?>
```

Ex2:
```
<?php

define("MESSAGE","Hello welcome to PHP",true);//not case sensitive

echo MESSAGE;
```

```
        echo message;
        ?>
```

Ex3:
```
<?php
  define("MESSAGE","Hello welcome to PHP",false);//case sensitive
  echo MESSAGE;
  echo message; ?>
```

Using const keyword:

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

Ex:
```
<?php
const MESSAGE="Hello welcome to PHP";
 echo MESSAGE;
 ?>
```

PHP Operators and Expressions:

- PHP Operator is a symbol i.e. used to perform operations on operands.
- In simple words, operators are used to perform operations on variables or values. For example:
- An *expression* is a bit of PHP that can be evaluated to produce a value.

EX: `$num=10+20;        //+ is the operator and 10,20 are operands`

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

PHP Operators can be categorized in following forms:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Conditional Operator

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

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

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

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

Comparison Operators :

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
!=	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

Incrementing and Decrementing Operators

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

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

Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

String Operators

There are two operators which are specifically designed for strings.

Operator	Description	Example	Result
.	Concatenation	$str1 . $str2	Concatenation of $str1 and $str2
.=	Concatenation assignment	$str1 .= $str2	Appends the $str2 to the $str1

Array Operators

The array operators are used to compare arrays:

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

Conditional Operator : There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation.

The conditional operator has this syntax

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Ex: Arithmetic Operators:

```
<!DOCTYPE html>
<html >
<head>
   <title>PHPArithmetic Operators</title>
</head>
<body>
<?php
$x = 10;
$y = 4;
echo($x + $y);
echo "<br>";
echo($x - $y);
echo "<br>";
echo($x * $y);
echo "<br>";
echo($x / $y);
echo "<br>";
echo($x % $y);
?>
</body>
</html>
```

Q) Flow-Control Statements in PHP:

- Control flow statements are used to control the flow of execution of statements.
- You can use *control flow statements* in your programs to conditionally execute statements, to repeatedly execute a block of statements.
- Generally, a program is executed sequentially, line by line, and a control structure allows you to alter that flow, usually depending on certain conditions.

Types:

1. Conditional Statements OR Decision Making Statements
2. Looping Statements or Iterative statements

PHP Decision Making or Conditional statements :

- In PHP conditional statements are the statements which are used to deciding the order of execution of statements based on certain conditions or perform different types of actions based on different actions.

In PHP we have the following conditional statements:

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

If Statement:

- The if keyword is used to check if an expression is true.
- If it is true, a statement is then executed.
- The statement can be a single statement or a compound statement.
- A compound statement consists of multiple statements enclosed by curly brackets.

Syntax : if (condition)
```
        {
           //code will be executed if specified condition is true
        }
```

Ex:
```
<html>
  <body>
     <?php
       $d = date("D");
            if ($d == "Fri")
          echo "Have a nice weekend!";
                 ?>
     </body>
</html>
```

If ... Else Statement:

- PHP if-else statement is executed whether condition is true or false.
- **If** you want to execute some code if a condition is true and another code if a condition is false, use the if... else statement.

Syntax : if (condition)
```
          {

          //code to be executed if condition is true;
          }
          else
```

```
 {
//code to be executed if condition is false;
}
```

Example

```
<html>
  <body>

    <?php
      $d = date("D");

      if ($d == "Fri")
        echo "Have a nice weekend!";

      else
        echo "Have a nice day!";
    ?>

  </body>
</html>
```

If then Else…If Statement

The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

Syntax

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

Ex:

```
<html>

  <body>
      <?php
      $d = date("D");
            if ($d == "Fri")
```

```
        echo "Have a nice weekend!";
            elseif ($d == "Sun")
        echo "Have a nice Sunday!";
            else
        echo "Have a nice day!";
    ?>
      </body>
</html>
```

Switch:

- PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.
- The switch statement works with two other keywords: case and break.
- The case keyword is used to test a label against a value from the round brackets.
- If the label equals to the value, the statement following the case is executed. The break keyword is used to jump out of the switch statement.

Syntax

```
switch(expression)
{
 case value1:
  //code to be executed
   break;
 case value2:
  //code to be executed
   break;
 ......
 default:
  code to be executed if all cases are not matched;
    }
```

Example:

```
<html>
  <body>

    <?php
      $d = date("D");

      switch ($d){
        case "Mon":
          echo "Today is Monday";
          break;
        case "Tue":
          echo "Today is Tuesday";
          break;

        case "Wed":
          echo "Today is Wednesday";
          break;

        case "Thu":
          echo "Today is Thursday";
          break;

        case "Fri":
          echo "Today is Friday";
          break;

        case "Sat":
          echo "Today is Saturday";
          break;

        case "Sun":
          echo "Today is Sunday";
          break;

        default:
          echo "Wonder which day is this ?";
      }
    ?>

  </body>
</html>
```

Q) Looping or iterative statements:

- Loops are used to execute the same block of code again and again, as long as a certain condition is met.
- Loops in PHP are used to execute the same block of code a specified number of times.

PHP supports following four loop types

- While Loop
- do While Loop
- for Loop
- for each loop

while Loop:

- The while loop executes a block of code as long as the specified condition is true.
- If the test condition is true then the code block will be executed. After the code has executed the test condition will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax : while (*condition is true*)

```
{
 code to be executed;
}
```

Ex:

```
<html l>
<head>
<title>PHP while Loop</title>
</head>
<body>
<?php
$i = 1;
while($i <= 3)
{
$i++;
 echo "The number is " . $i . "<br>";
}
  ?>
</body>
</html>
```

do…while loop statement:

- The do…while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.
- It executes the code at least one time always because condition is checked after executing the code.

Syntax :

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

Example:

```
<html >
<head>
 <title>PHP do-while Loop</title>
```

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

For loop:

- The PHP for loop allows the user to put all the loop-related statements (i.e. INITIALIZER; CONDITION; INCREMENTOR or DECREMENTOR) in one place. The structure is similar to C language.

Syntax

```
for ( initialization ; condition ; increment/decrement)
{
execute the statement;
}
```

initialize counter: Initialize the loop counter value.
test counter: Verify the loop counter whether the condition is true.
Increment/decrement counter: Increasing or decreasing the loop counter value.

EX:

```
<html >
<head>
  <title>PHP for Loop</title>
</head>
<body>
<?php
for($i=1; $i<=3; $i++){
echo "The number is " . $i . "<br>";
}
?>
</body>
</html>
```

PHP for each Loop:

The foreach loop is used to iterate over arrays.

Syntax:

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

Ex:

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

break Statement:

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

.**Syntax:** jump statement;

```
break;
```

Example:

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

Continue:

"Continue is used within looping structures to skip the rest of the current loop iteration" and continue execution. It is used to halt the current iteration of a loop but it does not terminate the loop.

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

Q) Code Blocks and Browser Output:

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

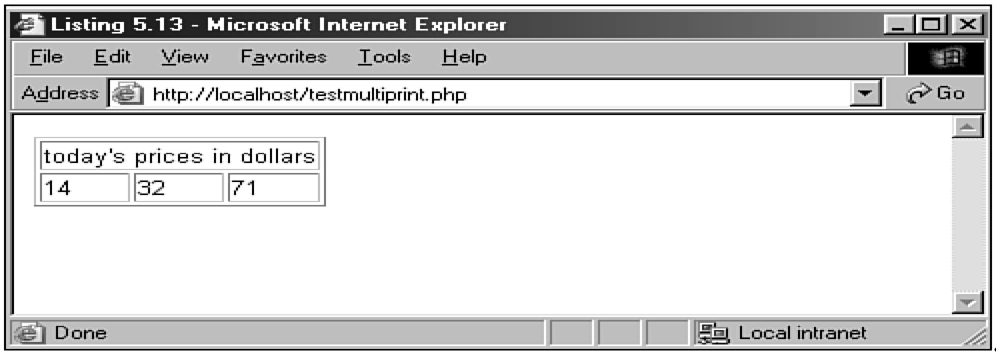
A Code Block Containing Multiple print() Statements

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

```
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.

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


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

today's prices in dollars		
14	32	71

.

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

Returning to HTML Mode Within a Code Block

```
 1: <html>
 2: <head>
 3: <title>table</title>
 4: </head>
 5: <body>
 6: <?php
 7: $displayprices = true;
 8: if ( $displayprices )
        {
 9: ?>
10:     <table border="1">
11:     <tr><td colspan="3">today's prices in dollars</td></tr>
12:     <tr><td>14</td><td>32</td><td>71</td>
13:     </table>
14: <?php
15: }
16: ?>
17: </body>
18: </html>
```

The important thing to note here is that the shift to HTML mode on line 9 only occurs if the condition of the if statement is fulfilled. This can save us the bother of escaping quotation marks and wrapping our output in print() statements. It might, however, affect the readability of our code in the long run, especially as our script grows larger.

Functions

Q) Functions:

- A Function in PHP is a reusable piece or block of code that performs a specific action.
- It takes input from the user in the form of parameters, performs certain actions, and gives the output.
- Functions can either return values when called or can simply perform an operation without returning any value.

Advantage of PHP Functions

Functions reduce the repetition of code within a program — Function allows you to extract commonly used block of code into a single component. Now you can perform the same task by calling this function wherever you want within your script without having to copy and paste the same block of code again and again.

Functions makes the code much easier to maintain — Since a function created once can be used many times, so any changes made inside a function automatically implemented at all the places without touching the several files.

Functions makes it easier to eliminate the errors — When the program is subdivided into functions, if any error occur you know exactly what function causing the error and where to find it. Therefore, fixing errors becomes much easier.

Functions can be reused in other application — Because a function is separated from the rest of the script, it's easy to reuse the same function in other applications just by including the php file containing those functions.

Creating PHP Function:

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

Functions Syntax:

1. Any name ending with an open and closed parenthesis is a function.
2. A function name always begins with the keyword *function*.
3. To call a function we just need to write its name followed by the parenthesis
4. A function name cannot start with a number. It can start with an alphabet or underscore.
5. A function name is not case-sensitive.

Syntax : **function*functionName*()**

```
{
code to be executed;
 }
```

<u>Calling a Function:</u>

To call a function in php we use the name of the function ended by semicolon;

syntax : functionname();

Example:

```
<html>
 <head>
 <title>Writing PHP Function</title>
 </head>
  <body>
<?php
 function  Message()
{
  echo " Have a nice day";
 }
Message();
Message();
 ?>
  </body>
</html>
```

Result: Have a nice day

<u>Function With Parameters or Arguments:</u>

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

Syntax::function functionname($firstparameter, $secondparameter)

```
{
   executable code;
}
```

EX:<html>
```
<head>
 <title>Writing PHP Function with Parameters</title>
</head>
<body>

            <?php
           function addFunction($num1, $num2,$num3)
            {
           $sum = $num1 + $num2;
            echo "Sum of the two numbers is : $sum";
              }
             addFunction(10, 20);
              ?>
           </body>
           </html>
```

result –Sum of the two numbers is : 30

Call By Reference

- It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.
- Any changes made to an argument in these cases will change the value of the original variable.
- By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

EX: functionref.php
```
      <html>
       <head>
       <title>Writing PHP call by reference</title>
        </head>
         <body>
    <?php
    function adder(&$str2)
    {
       $str2 .= 'Call By Reference';
    }
    $str = 'Hello ';
    adder($str);
    echo $str;
    ?>
       </body>
       </html>
```

Output: Hello Call By Reference

Function with Default Arguments:

- We can specify a default argument value in function.
- While calling PHP function if you don't specify any argument, it will take the default argument.

File: functiondefaultarg.php

```
        <html>
         <head>
         <title>Writing PHP Function with default arguments</title>
          </head>
           <body>
      <?php
      function sayHello($name="lakshmi")
      {
      echo "Hello $name<br/>";
      }
      sayHello("Rajesh");
      sayHello();
      sayHello("John");
  ?>
        </body>
        </html>
```

Output: Hello Rajesh

```
        Hello mahesh
        Hello John
```

Function With Returning Value

- Functions can also return values to the part of program from where it is called.
- The *return* keyword is used to return value back to the part of program, from where it was called.

ex:functiondefaultarg.php

```
        <html>
         <head>
         <title>Writing PHP Function with returning value</title>
          </head>
          <body>
              <?php
              function cube($n){
              return $n*$n*$n;
              }
              echo "Cube of 3 is: ".cube(3);
              ?>
        </body>
        </html>
```

Output: Cube of 3 is: 27

PHP Variable Length Argument Function

- PHP supports variable length argument function.
- It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

Ex:

```
<html>
 <head>
 <title>Writing PHP Function with Variable length argument</title>
  </head>
 <body>
<?php
 function add(...$numbers) {
 $sum = 0;
foreach ($numbers as $n) {
    $sum += $n;
   }
return $sum;
 }
     echo add(1, 2, 3, 4);
 ?>
</BODY>
</HTML>
```

Recursive Function

- PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.
- It is recommended to avoid recursive function call over 200 recursion level because it may smash the stack and may cause the termination of script.

Example 1: Printing number

```
   <html>
 <head>
 <title>Writing PHP recursive function</title>
  </head>
 <body>
<?php
  function display($number)
 {
    if($number<=5)
{
    echo "$number <br/>";
  display($number+1);
  }
  }
display(1);
 ?>    </body></html>
```

Arrays

Q) Arrays:

- Arrays in PHP are a type of data structure that allows us to store multiple elements of similar data type under a single variable.
- The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key.
- For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.
- An array can hold any number of values, including no values at all.
- Each value in an array is called an element.
- You access each element via its index, which is a numeric or string value. Every element in an array has its own unique index.
- An element can store any type of value, such as an integer, a string, or a Boolean. You can mix types within an array — for example, the first element can contain an integer, the second can contain a string, and so on.
- An array's length is the number of elements in the array.

create an array in PHP

- An array is created using an array() function in PHP.

Syntax: $myArray = array(values);

Types of Arrays:

There are 3 types of array in PHP.

1. Numerical Array or Indexed Array
2. Associative Array
3. Multidimensional Array

Numerical Arrays:

- These arrays can store numbers, strings and any object but their index will be represented by numbers.
- By default, the index starts at zero.
- All PHP array elements are assigned to an index number by default.

These arrays can be created in two different ways.

There are two ways to define indexed array:

1st way: $colors = array("Red", "Green", "Blue");

2nd way:
```
<?php
$colors[0] = "Red";
$colors[1] = "Green";
$colors[2] = "Blue";
?>
```

Example

```
<html>
   <body>
      <?php
       $numbers = array( 1, 2, 3, 4, 5);
            foreach( $numbers as $value )
         {
         echo "Value is $value <br />";
       }
          /* Second method to create array. */
       $numbers[0] = "one";
       $numbers[1] = "two";
       $numbers[2] = "three";
       $numbers[3] = "four";
       $numbers[4] = "five";
             foreach( $numbers as $value )
         {
         echo "Value is $value <br />";
         }
     ?>
       </body>
</html>
```

result –

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

Associative Arrays

- An array with a string index where instead of linear storage, each value can be assigned a specific key.
- Associative array will have their index as string so that you can establish a strong association between key and values.
- To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.
- We can associate name with each array elements in PHP using => symbol.

Syntax : $variablename['key_name'] = value;

or

$variable_name = array('keyname' => value);

- "$variable_name..." is the name of the variable
- "['key_name']" is the access index number of the element
- "value" is the value assigned to the array element.

Example: $age=array("samivulla"=>"21", "raviteja"=>"20", "Rajesh"=>"19");

or

```
$age['samivulla'] = "21";
$age['raviteja'] = "20";
$age['rajesh'] = "19"
```

Ex: <?php

```
$salary=array("Samivulla"=>"350000","venkat"=>"450000","Kartik"=>"200000");
echo "Samivulla salary: ".$salary["samivulla"]."<br/>";
echo "Venkat salary: ".$salary[vekat"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";  ?>
```

Output:

```
Samivulla salary: 350000
venkatsalary: 450000
Kartiksalary: 200000
```

MultidimensionalArrays

- 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.

Syntax:

```
$arrayname=array(

        array(value1,value2,value3),

        array(value1,value2,value3),

        array(value1,value2,value3)

);
```

Ex:

```
<?php
$emp = array
                (

                array(1,"Samivulla",350000),
                array(2,"venkat",450000),
                array(3,"karthik",200000)
                 );
for ($row = 0; $row < 3; $row++)
 {
  for ($col = 0; $col < 3; $col++)
 {
    echo $emp[$row][$col]."  ";
  }
  echo "<br/>";
}
?>
```

Q) **Functions of Array:**

1.array() Function: The array() function is used to create an array.

Syntax: array(*value1, value2, value3, etc.*)

Ex: <?php

$names=array("lakshmi","anusha","aparna");

echo "$names[0]<br>$names[1]<br>$names[2]";

?>

Output: Sireesha

Anusha

aparna

2. count() Function: The count() function returns the number of elements in an array.
Syntax : count(array);

Ex: <?php

$names=array("lakshmi","anusha","aparna");

echo count($names);

?>

Output: 3

3.sort() Function: The sort() function sorts an array in ascending order.

Syntax :sort(array);

Ex: <?php

```
        $names=array("lakshmi","anusha","aparna","ramesh","suresh");
         sort($names);
        $length=count($names);
        for($x=0;$x<$length;$x++)
          {
         echo $names[$x];
```

```
 echo "<br>";
  }
?>
```

Output: anusha

```
aparna
lakshmi
ramesh
suresh
```

4. **array_push() Function:**
The array_push() function inserts one or more elements to the end of an array.
Syntax : array_push(array, value1, value2, ...);
Ex: <?php
```
$names=array("lakshmi","anusha","aparna");
array_push($names,"janaki","amrutha");
print_r($names)
?>
```

5. **array_pop() Function:**
The array_pop() function deletes the last element of an array.
Syntax : array_pop(array);
Ex: <?php
```
$names=array("lakshmi","anusha","aparna");
array_pop($names);
print_r($names)
?>
```

6. **array_reverse() Function:**
The array_reverse() function returns an array in the reverse order.
Syntax : array_reverse(array);
Ex: <?php
```
$names=array("lakshmi","anusha","aparna");
array_reverse($names);
print_r($names);
?>
```

7. **array_replace() Function:**
It Replace the values of the first array ($a1) with the values from the second array ($a2).
Syntax: array_replace(array1, array2, array3, ...);

Ex: <?php
$names1=array("lakshmi","anusha","aparna");
$names2=array("ram","venkat","kishore");
print_r(array_replace($names1,$names2));
?>

8. **array_shift() Function:**
   The array_shift() function removes the first element from an array, and returns the value of the removed element.
   Syntax : array_shift(array);

Ex:
```
<?php
$names=array("jayasree","lakshmi","anusha","aparna");
array_shift($names);
print_r($names);
?>
```

9. **array_unshift() Function:**
   The array_unshift() function inserts new elements to an array. The new array values will be inserted in the beginning of the array**.**
   Syntax : array_unshift(array, value1, value2, value3, ...);

Ex:
```
<?php
$names=array("jayasree","lakshmi","anusha","aparna");
array_unshift($names,"ram");
print_r($names);
?>
```

10. **array_slice() Function:**
    The array_slice() function returns selected parts of an array.
    Syntax : array_slice(array, start, length);

Ex:
```
<?php
$names=array("jayasree","lakshmi","anusha","aparna","ram","suresh","venk
at");
print_r(array_slice($names,2,4));
?>
```

11. **array_splice() Function:**
    The array_splice() function removes selected elements from an array and replaces it with new elements. The function also returns an array with the removed elements.

    Syntax :array_splice(array, start, length, array)

Ex:
```
<?php
$names1=array("jayasree","lakshmi","anusha","aparna","ram");
```

```
$names2=array("venkat","kishore","ramesh");
array_splice($names1,0,2,$names2);
print_r($names1)
?>
```

12. array_search() Function:

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

Syntax : array_search(value, array);

Ex: <?php
```
$names=array("jayasree","lakshmi","anusha","aparna","ram");
echo array_search("lakshmi",$names);
?>
```

13. array_sum() Function:

The array_sum() function returns the sum of all the values in the array.

Syntax: array_sum(array);

Ex: <?php
```
$a=array(5,15,25);
echo array_sum($a);
?>
```

14. <u>PHP array_intersect() function</u>

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

Syntax:

array array_intersect (array $array1 , array $array2 [, 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 />";
}
?>
```

15. rsort() Function:

The rsort() function sorts the values of the indexed array in descending order.

Syntax

rsort(array);

```
<?php
$fruits = array("apple", "orange", "mango", "banana", "kiwi");
rsort($fruits);
print_r($fruits);?>
```