UNIT – 1

1.0 Introduction to PHP

1.1 What is PHP?

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

1.2 What is a PHP File?

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

1.3 What Can PHP Do?

1. PHP can generate dynamic page content
2. PHP can create, open, read, write, delete, and close files on the server
3. PHP can collect form data
4. PHP can send and receive cookies
5. PHP can add, delete, modify data in your database
6. PHP can be used to control user-access
7. PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

1.4 Why PHP?

1. PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
2. PHP is compatible with almost all servers used today (Apache, IIS, etc.)
3. PHP supports a wide range of databases
4. PHP is free.
5. PHP is easy to learn and runs efficiently on the server side

2.0 History of PHP

1. PHP 1 [PHP: Hypertext Preprocessor] was created by RasmusLerdorf in 1994. It was initially developed for HTTP usage logging and server-side form generation in Unix.

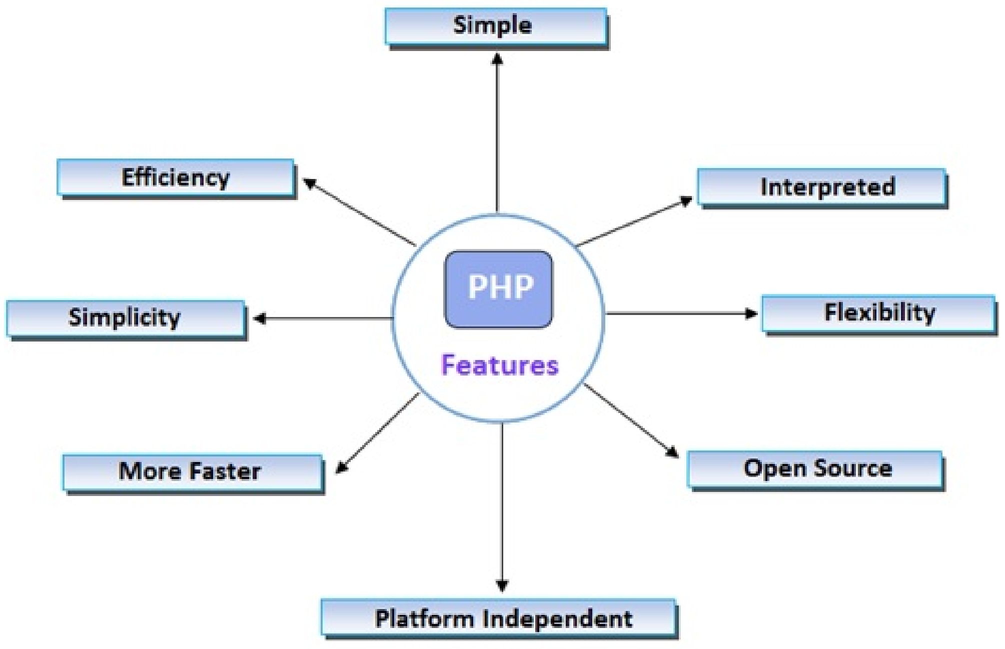
2. PHP 2 (1995) transformed the language into a Server-side embedded scripting language. Added database support, file uploads, variables, arrays, recursive functions, conditionals, iteration, regular expressions, etc.

3. PHP 3 [1998] added support for ODBC data sources, multiple platform support, email protocols, and new parser written by ZeevSuraski .

4. PHP 4 (2000) became an independent component of the web server for added efficiency. The parser was renamed the Zend Engine. Many security features were added.

5. PHP 5 (2004) adds Zend Engine II with object oriented programming robust XML support using the libxrnl2 library, SQLite has been bundled with PHP.

3.0 Features of PHP



Source: http://www.tracesoftware.in/phps-main-features/

It is most popular and frequently used worldwide scripting language, the main reason of popularity is; It is open source and very simple.

1. Simple and Faster
2. Interpreted and Open Source
3. Case Sensitive and Simplicity
4. Efficiency
5. Platform Independent
6. Security, Familiarity and Flexibility

(1) **Simple, Familiar and ease of use:** Its popularly known for its simplicity, familiarity and easy to learn the language as the syntax is similar to that of 'C' or Pascal language. So the language is very logical and well organized general-purpose programming language. Even people with a normal programming background can easily understand and capture the use of language. PHP is very advantageous for new users as it's a very reliable, fluent, organized, clean, demandable and efficient.

The main strength of PHP is the availability of rich pre-defined functions. The core distribution helps the developers implement dynamic websites very easily with secured data. PHP applications are very easy to optimize.

(2) **Flexibility:** PHP is known for its flexibility and embedded nature as it can be well integrated with **HTML**, **XML**, **JavaScript** and many more. PHP can run on multiple operating systems like **Windows**, **Unix**, **Mac OS**, **Linux**, **etc**. The PHP scripts can easily run on any device like laptops, mobiles, tablets, and computer. It is very comfortably integrated with various Databases. Desktop applications are created using advanced PHP features. The executable PHP can also be run on command-line as well as directly on the machine. Heavyweight applications can be created without a server or browser.It also acts as an excellent interface with relational databases.

(3) **Open Source:** All PHP frameworks are open sources, no payment is required for the users and its completely free. User can just download PHP and start using for their applications or projects. Even in companies, the total cost is reduced for software development providing morereliability and flexibility.It supports a popular range of databases like MySQL, SQLite, Oracle, Sybase,Informix, and PostgreSQL.

(4) PHP provides libraries to access these databases to interact with web servers. Developers are free to post errors, inspect codes and can contribute to code as well as bug fixing. Many frameworks like Codeignitor, Zend Framework, CakePHP make use of PHP.Even many popular content management systems like WordPress, Joomla and Drupal use PHP as prime language.

Because of the above reasons many web hosting companies and Internet Service providers prefers PHP.

(5) **Cross-platform compatibility:** PHP is multi-platform and known for its portability as it can run on any operating System and windows environments. The most common are XAMPP (**Windows**, **ApacheServer**, **MySQL**, **Perl**, and **PHP**) and LAMP (**Linux**, **Apache**, **MySQL**, **PHP**). As PHP is platform-independent, it's very easy to integrate with various databases and other technologies without re-implementation. It effectively saves a lot of energy, time and money.

(6) **Fast and efficient performance:** Users generally prefer fast loading websites. For any web development, speed becomes an important aspect which is taken care of by PHP.

(7) PHP scripts are faster than other scripting languages like **ASP.NET**, **PERL**, and **JSP**. The memory manager of PHP 7 is very optimized and fast as compared to older versions of PHP.

Even connecting to the database and loading of required data from tables, are faster than other programming languages. It provides a built-in module for easy and efficient database management system. The high speed of PHP is advantageous for users for its server administration and mail functionality. Also, it supports session management and removing of unwanted memory allocation.

4.0 Merits and Demerits of PHP

Merits

1. PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
2. PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
3. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
4. PHP is pleasingly(satisfying) zippy(fresh) in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
5. PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
6. PHP is forgiving(understanding): PHP language tries to be as forgiving as possible. PHP Syntax is C-Like.

Demerits

1. **Security:** Since it is open sourced, 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 it.
2. **Not suitable of large applications:** It will be difficult to use it for programming huge applications. Since the programming language is not highly modular, huge applications created out of the programming language will be difficult to maintain.
3. **Weak type:** Implicit conversion may surprise unwary programmers and lead to unexpected bugs. Confusion between arrays and hash tables. This is slow and could be faster. There are often a few ways to accomplish a task. It is not strongly typed. It is interpreted and uses curly braces.
4. **Poor Error Handling Method:** The framework has a bad error handling method. It is not a proper solution for the developers. Therefore, as a qualified PHP developer, you will have to overcome it.
5. **PHP is unable to handle large number of apps:** The technology is helpless to support a bunch of apps. It is highly tough to manage because, it is not competent modular. It already imitates the features of Java language.

4.1 Applications of PHP

1. PHP performs system functions, i.e. from files on a system it can create, open, read, write,

and close them.

2. PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
3. You add, delete, modify elements within your database through PHP.
4. Access cookies variables and set cookies.
5. Using PHP, you can restrict users to access some pages of your website.
6. It can encrypt data.

5.0 General structure of PHP

PHP code is always separated from HTML code with symbols
<?php (LESS-THAN sign followed by question mark and the word php) and
?> (a question mark followed by a GREATER-THAN sign).

<?phpindicates the start of a PHP block or code and tells the web server thatall statements that follow until **?>** are PHP statements.

```
<?php

//<some valid PHP syntax>

?>
```

6.0 Displaying Output

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print

PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command.

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

PHP print Statement

The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command.

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

7.0 Escaping Special Characters

Escaping to PHP:

The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP'.

1. **Canonical PHP tags: <?php … ?>**
   If you use this style, you can be positive that your tags will always be correctly interpreted.

2. **Short-open (SGML-style) tags: <? … ?>**
   Short tags are, as one might expect, the shortest option You must do one of two things to enable PHP to recognize the tags −
   Choose the --enable-short-tags configuration option when you're building PHP.
   Set the short_open_tag setting in your php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags.

3. **ASP-style tags: <% … %>**
   ASP-style tags mimic(copy) the tags used by Active Server Pages to delineate(describe) code blocks.

4. **HTML script tags: <script language= "PHP"> …. </script>**

The escape-sequence replacements are −

\n is replaced by the newline character
\r is replaced by the carriage-return character
\t is replaced by the tab character
\$ is replaced by the dollar sign itself ($)
\" is replaced by a single double-quote (")
\\ is replaced by a single backslash (\)

8.0 Comments

A comment is the portion of a program that exists only for the human reader and stripped out(take away from) before displaying the programs result. There are two commenting formats in PHP −

1. Single Line comments:

They are generally used for short explanations or notes relevant to the local code. Here are theexamples of single line comments.

```
<?
  # This is a comment, and
  # This is the second line of the comment

  // This is a comment too. Each style comments only
  print "An example with single line comments";
?>
```

2. Multi-lines comments:

They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here is the example of multi lines comments.

```
<?
  /* This is a comment with multiline
Author:Ivan Bayross
    Purpose: Multiline Comments Demo
    Subject: PHP
  */
    print "An example with multi line comments";
?>
```

9.0 Variables

The main way to store information in the middle of a PHP program is by using a variable.
Here are the most important things to know about variables in PHP.

1. All variables in PHP are denoted with a leading dollar sign ($).
2. The value of a variable is the value of its most recent assignment.
3. Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
4. Variables can, but do not need, to be declared before assignment.
5. Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
6. Variables used before they are assigned have default values.
7. PHP does a good job of automatically converting types from one to another when necessary.
8. PHP variables are Perl-like.

Variable Scope:

Scope can be defined as the range of availability a variable has to the program in
which it is declared. PHP variables can be one of four scope types −

1. Local Variables
2. Function Parameters
3. Global Variables
4. Static Variables

Rules for PHP variables:

1. A variable start with the $ sign, followed by the name of the variable.
2. A variable name must start with a letter or the underscore character.
3. A variable name cannot start with a number.
4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
5. Variable names are case-sensitive ($age and $AGE are two different variables)

Declaring and Assign PHP Variables

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

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Destroy PHP Variables

Unset () Description

unset (mixed $var , mixed ...$vars) : void

unset () destroys the specified variables. The behavior of unset () inside of a function can vary depending on what type of variable you are attempting to destroy. If a globalized variable is unset () inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset () was called.

```
<?php
function destroy_foo()
{  global $foo;
   unset($foo); }
$foo = 'bar';
destroy_foo();
echo $foo;
?>
```

10.0 Data Types

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object

- NULL
- Resource

PHP String

A string is a sequence of characters, like "Hello world!".
A string can be any text inside quotes. You can use single or double quotes:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.
In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

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

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

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

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

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Variables can also be emptied by setting the value to NULL:

```
<?php
$x = "Hello world!";
$x = null;
```

```
var_dump($x);
?>
```

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

We will not talk about the resource type here, since it is an advanced topic.

11.0 PHP Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Create a PHP Constant

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

Syntax

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

Parameters:

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

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

PHP Constant Arrays

```
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
```

Constants are Global

Constants are automatically global and can be used across the entire script.

This example uses a constant inside a function, even if it is defined outside the function:

```
<?php
```

```
define("GREETING", "Welcome to W3Schools.com!");
function myTest() {
  echo GREETING;
}
myTest();
?>
```

12.0 Operators

Operators are used to perform operations on variables and values.

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

PHP Arithmetic Operators

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

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

PHP Assignment Operators

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

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction

x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

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

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

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

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

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

--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

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

PHP String Operators

PHP has two operators that are specially designed for strings.

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

PHP Concatenation Operators

PHP Compensation Operator is used to combine character strings.

Operator	Description
.	The PHP concatenation operator (.) is used to combine two string values to create one string.
.=	Concatenation assignment.

```
<?php
   $name="Karamchand";
   $lastName="Gandhi";
   echo $name." ".$lastName; // Outputs Karamchand Gandhi
   $a="Hello";
   $a .= " Gandhi!";
   echo $a; // Outputs Hello Gandhi!
?>
```

13.0 Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET

(1) PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

```
<?php
$x = 75;
$y = 25;
 function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z;
?>
```

(2) PHP $_SERVER

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

(3) PHP $_REQUEST

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

```
<html>
<body>

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

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

(4) PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

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

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

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

(5) PHP $_GET

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

The example below shows the code in "test_get.php":

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>
</body>
</html>
```

UNIT – 2

Topics Covered:

1. Control structures
2. Looping structures
3. 1-D Array & its manipulation (Storing Data, Assigning, Accessing Array Elements, Displaying)
4. User-Defined Functions
5. Function Scope

Decision-making statements:

- **if** statement - use this statement to execute some code only if a specified condition is true.
- **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false.
- **if...elseif...else** statement - use this statement to select one of several blocks of code to be executed.
- **switch** statement - use this statement to select one of many blocks of code to be executed.

1. if statement

The **if** statement only executes any code if the condition is true. If the condition isn't met, then the code is completely ignored.

```
if (condition)
{
        Execute a line of code;
}
```

Example:

```
<?php
        $age = 50;
        if ($age > 30)
        {
                echo "Your age is greater than 30!";
        }
?>
```

Note: if there is only single executable statement inside **if** condition then rewrite above code without brackets.

2. The if...else Statement

The **if...else** statement executes some code if a condition is true and another code if that condition is false.

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

Example:

```
<?php
	$age = 50;
	if ($age < 30)
	{
		echo "Your age is less than 30!";
	}
	else
	{
		echo "Your age is greater than or equal to 30!";
	}
?>
```

3. The if...elseif...else Statement

The **if...elseif...else** statement executes different codes for more than two conditions.

```
if (condition)
{
	code to be executed if this condition is true;
}
elseif (condition)
{
	code to be executed if first condition is false and this condition is true;
}
else
{
	code to be executed if all conditions are false;
}
```

Example

```
<?php
	$age = 50;
	if ($age < 30)
	{
```

```
                echo "Your age is less than 30!";
        }
        elseif ($age > 30 && $age < 40)
        {
                echo "Your age is between 30 and 40!";
        }
        elseif ($age > 40 && $age < 50)
        {
                echo "Your age is between 40 and 50!";
        }
        else
        {
                echo "Your age is greater than 50!";
        }
?>
```

4. The switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

```
        switch (n)
        {
                case label1:
                        code to be executed if n=label1;
                        break;
                case label2:
                        code to be executed if n=label2;
                        break;
                case label3:
                        code to be executed if n=label3;
                        break;

                          ...
                default:
                        code to be executed if n is different from all labels;
        }
```

First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure.If there is a match, the block of code associated with that **case** is executed.

Use break to prevent the code from running into the next case automatically. The default statement is used if **no match is found**.

Example :

```
<?php
        $favcolor = "red";
        switch ($favcolor)
        {
                case "red":
                        echo "Value of i is red!";
                        break;
                case "blue":
                        echo "Value of i is blue!";
                        break;
                case "green":
                        echo "Value of i is green!";
                        break;
                default:
                        echo " Value of i is neither red, blue, nor green!";
        }
?>
```

Looping statements:

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

1.While Loop

The **while** loop - Loops through a block of code as long as the specified condition is true.

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

Examples

```
<?php
        $x = 1;

        while($x <= 5)                          // displays the numbers from 1 to 5:
        {
                echo "The number is: $x <br>";
                $x++;
```

```
    }
?>
```

2. do while Loop

The **do...while** loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

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

The example below first sets a variable $x to 1 ($x = 1). Then, the **do while** loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
    $x = 1;
    do
    {
        echo "The number is: $x <br>";
        $x++;
    } while ($x <= 5);
?>
```

Note: In a **do...while** loop the condition is tested AFTER executing the statements within the loop. This means that the **do...while** loop will execute its statements at least once, even if the condition is false.

3. for Loop

The **for** loop is used when you know in advance how many times the script should run.

```
    for (init counter; test condition; increment counter)
    {
        code to be executed for each iteration;
    }
```

Parameters:

init counter: Initialize the loop counter value

test condition: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

Examples

```
<?php
	for ($x = 0; $x <= 10; $x++)			// displays the numbers from 0 to 10
	{
		echo "The number is: $x <br>";
	}
?>
```

4. foreach Loop

The **foreach** loop works only on arrays, and is used to loop through each key/value pair in an array.

```
	foreach ($array as $value)
	{
		code to be executed;
	}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.The following example will output the values of the given array ($subject):

```
<?php
	$subject = array("Java", "Php", ".Net", "Python");

	foreach ($subject as $value)
	{
	echo "$value <br>";
	}
?>
```

Output

```
	Java
	Php
	.Net
	Python
```

The following example will output both the keys and the values of the given array ($age):

```
<?php
	$age = array("Reyash"=>"35", "Sem"=>"37", "Yug"=>"43");
	foreach($age as $x => $val)
	{
		echo "$x = $val<br>";
	}
?>
```

Output

```
	Reyansh = 35
	Sem = 37
```

```
Yug = 43
```

The break statement

The **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 next statement to the loop will be executed.

Example

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

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

Output

```
Loop stopped at i = 3
```

The continue statement

The **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.

In the following example loop prints the value of array but for value=3,condition becomes true, and it skips the code and next value is printed.

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

Output

```
Value is 1
Value is 2
Value is 4
Value is 5
```

PHP Arrays

An array stores multiple values under a single variable name. The **array()** function is used to create an array:

array();

There are three types of arrays:

- **Numeric/Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

1. Numeric/Indexed Arrays

There are two ways to create indexed arrays and store/assign values in an array:

General form : **$arrayname = array(list of values separated by comma);**

The index can be assigned automatically (index always starts at 0), like this:

```
/* First method to create array. */
    $numbers = array( 1, 2, 3, 4, 5);
```

Or the index can be assigned manually:

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

The following example creates an indexed array named $subject, assigns three elements to it, and then prints a text containing the array values:

Accessing Array elements:

General form: **$arraname[index];**

Example :

```
<?php
```

```
        $subject = array("Java", "Php", ".Net");
        echo "I like " . $subject[0] . ", " . $subject[1] . " and " . $ subject[2] . ".";
?>
```

Get The Length of an Array - The count() Function:

The count() function is used to return the length (the number of elements) of an array:

```
<?php
        $subject = array("Java", "Php", ".Net");
        echo count($subject);
?>
```

Loop Through an Indexed Array:

To loop through and print all the values of an indexed array, you could use a **for** loop, like this:

```
<?php
        $subject = array("Java", "Php", ".Net");
        $arrlength = count($subject);
        for($x = 0; $x < $arrlength; $x++)
        {
                echo $subject[$x];
                echo "<br>";
        }
?>
```

Output:

```
        Java
        Php
        .Net
```

2. Associative Arrays

An associative array, each ID key is associated with a value. When storing data about specific named values, a numerical array is not always the best way to do it.

Example 1 - First method to create associative array.

In this example we use an array to assign subject name like PHP, Computer Network and Maths to the different keys like P, C and M respectively:

$arrayname = array(key1=>value1, key2=>value2,…, keyN=>valueN);

e.g. $subject = array("P"=>"PHP", "C"=>"Computer Networks", "M"=>"Maths");

Example 2- Second method to create associative array

This example is the same as example 1, but shows a different way of creating the array:

```
        $arrayname [key1] = value1;
        $arrayname [key2] = value2;
e.g.    $subject['P'] = "PHP";
        $subject['C'] = "Computer Networks";
```

The ID keys can be used in a script:

```
<?php
        $subject['P'] = "PHP";
        $subject['C'] = "Computer Networks";
        echo "My favourite subject is " . $subject['P'] . ";
?>
```

Output:

```
        My favourite subject is PHP.
```

User-Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.A function is a block of statements that can be used repeatedly in a program.A function will not execute automatically when a page loads.A function will be executed by a call to the function. A user-defined function declaration starts with the word function:

```
        function function_name()
        {
                code to be executed;
        }
```

Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

Example :

```
        <?php
                function greetings()
                {
                        echo "Hello world!";
                }
                greetings ();           // call the function
        ?>
```

Output:

```
        Hello World!
```

Function With Arguments

In following example, a function is defined with two formal arguments

```
        <?php
```

```
                function add($arg1, $arg2)
                {
                        echo $arg1+$arg2 . "\n";
                }
                add(10,20);
        ?>
```

Output:

```
        30
```

Function Return

User defined function in following example processes the provided arguments and returns a value to calling environment. In second call, even if arguments are string, PHP converts them into integer and performs addition. (Implicit type conversion)

```
<?php
        function add($arg1, $arg2)
        {
                return $arg1+$arg2;
        }
        $val=add(10,20);
        echo "addition:". $val. "\n";
        $val=add("10","20");
        echo "addition: $val";
?>
```

Output:

```
        addition: 30
        addition: 30
```

Function With Default Argument Value

While defining a function, a default value of argument may be assigned. If value is not assigned to such argument while calling the function, this default will be used for processing inside function.

In following example, a function is defined with argument having default value. In second call, function is called without passing value. In this case, user argument takes its default value.

```
<?php
        function welcome($user="Guest")
        {
                echo "Hello $user\n";
        }
        //overrides default
        welcome("admin");
        //uses default
        welcome();
?>
```

Output

```
        Hello admin
        Hello Guest
```

<u>Function Scope</u>

PHP Variables Scope

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 used. PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?php
        $x = 5;                         // global scope
        function test()
        {
                                        // using x inside this function will generate an error
                echo "<p>Variable x inside function is: $x</p>";
        }
        test();
        echo "<p>Variable x outside function is: $x</p>";
?>
```

Output:

```
        Variable x inside function is:
        Variable x outside function is: 5
```

Variable with local scope:

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

```
<?php
        function myTest()
        {
                $x = 5;                                 // local scope
                echo "<p>Variable x inside function is: $x</p>";
        }
        myTest();
                                        // using x outside the function will generate an error
        echo "<p>Variable x outside function is: $x</p>";
?>
```

Output:

```
        Variable x inside function is: 5
        Variable x outside function is:
```

PHP The global Keyword

The **global** keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function) as below. (Alternatively use $GLOBALS)

```
<?php
        $x = 5;
        $y = 10;
        function myTest()
        {
                global $x, $y;
                $y = $x + $y;
        }
        myTest();
        echo $y; // outputs 15
?>
```

Output:

```
        15
```

PHP The static Keyword

Normally, when a function is completed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.To do this, use the **static** keyword when you first declare the variable:

```
<?php
        function myTest()
        {
                static $x = 0;
                echo $x;
                $x++;
        }
        myTest();
        myTest();
?>
```

Output

```
0
1
```

UNIT – 3

Working with Numbers

1.abs()

It is the most basic function and returns the absolute value of the parameter passed to it. i.e. the function leaves positive values untouched, while converts negative values into positive values

Syntax: **abs(number value)**

e.g.:

```
<?php
	echoabs(-3.1);
	echo "<br>";
	echoabs(47.1);
?>
```

Output:

```
	3.1
	47.1
```

2.ceil()

It accepts one parameter, a number and returns the round to the nearest integer greater than the number.

Syntax: **ceil(number value)**

e.g.:

```
<?php
	echoceil(15.85);
	echo "<br>";
	echoceil(47.1);
?>
```

Output:

```
	16
	48
```

3.floor()

It accepts one parameter, a number and returns the round to the nearest integer smaller than the number.

Syntax: **floor(number value)**

e.g.:

```
<?php
        echofloor(15.85);
        echo "<br>";
        echofloor(47.1);
?>
```

Output:

```
        15
        47
```

4.round()

It accepts two parameters, a number and returns the round to the nearest integer smaller than the number.

Syntax: **round(number value [,int precision])**

e.g.:

```
<?php
        echoround(15.85);
        echo "<br>";
        echoround(15.85,1);
?>
```

Output:

```
        16
        15.9
```

5.sqrt()

It is short name of square root and takes just one parameter i.e. the value whose square root needs to be calculated.

Syntax: **sqrt(number value)**

e.g.:

```
<?php
        echosqrt(16);
        echo "<br>";
        echosqrt(9);
?>
```

Output:

```
        4
        3
```

6.pow()

It takes two parameters i.e. a base and a power to raise it by. The value in the first parameter is multiplied with itself, one time less than the value entered as the second parameter. The pow()

function is capable of handling negative powers for the second parameter.

Syntax: **pow(number base_value, number exponent)**

e.g.:

```
<?php
	echopow(3,2);
	echo “<br>”;
	echopow(4,3);
?>
```

Output:

```
	9
	64
```

Working with Strings

1. Getting length of a String

strlen() displays the length of any string. It is more commonly used in validating input fields where the user is limited to enter a fixed length of characters.

Syntax:strlen(string);

e.g.:

```
<?php
	echostrlen(“Welcome”);
?>
```

Output:

```
7
```

2. Reversing a String

strrev() is used for reversing a string. You can use this function to get the reverse version of any string.

Syntax: strrev(string)

e.g.:

```
<?php
	echostrrev(“TYBCA”);
?>
```

Output:

```
ACBYT
```

3. Finding Position of Text Within a String

strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return “False”. Strops() is most commonly used in validating input fields like email. It is case-sensitive search.

Syntax: strpos(string,text);

e.g.:
```
<?php
	echostrpos(“Welcome to Cloudways”,”Cloudways”);
?>
```
Output:
```
11
```

4. Replacing text within a string

str_replace() is a built-in function, basically used for replacing specific text within a string. It is case-sensitive searching.
Syntax: str_replace(string to be replaced, replace with,string)

e.g.:
```
<?php
	echostr_replace(“SPU”, “PHP world”, “Welcome to SPU”);
?>
```
Output:
```
Welcome to PHP world
```

5. Converting whole string to UPPERCASE

strtoupper() is used to convert the whole string into uppercase. The lowercase alphabets are converted to uppercase.
Syntax: strtoupper(string);

e.g.:
```
<?php
	echostrtoupper(“welcome to SPU”);
?>
```
Output:
```
WELCOME TO SPU
```

6. Converting whole string to lowercase

strtolower() is used to convert a string into lowercase. The uppercase alphabets are converted to lowercase.
Syntax: strtolower(string);

e.g.:
```
<?php
	echostrtolower(“WELCOME TO SPU”);
?>
```
Output:
```
welcome to spu
```

7. Repeating a String

str_repeat()is used for repeating a string a specific number of times.
Syntax: str_repeat(string,repeat)

e.g.:
```
<?php
        echostr_repeat("*",3);
?>
```
Output:
```
***
```

8. Displaying part of String

Through substr() function you can display or extract a string from a particular position. If the length is not mentioned then it extracts from start position till the end of the string.
Syntax: substr(string,start,length)

e.g.:
```
<?php
        echosubstr("Welcome to SPU",5)."<br>";
        echosubstr("Welcome to SPU",0,6)."<br>";
?>
```
Output:
```
me to SPU
Welcom
```

9. Removing white spaces from a String

Trim() is used to remove white spaces as well as predefined characters from both the sides of a string.
Syntax: trim(string) or trim(string[, charlist])

e.g.:
```
<?php
$str = "   Testing   ";          // 3 spaces before and 3 spaces after the string are there
echo"*".trim("$str")."*";
?>
```
Output:
```
*Testing*
```

<u>Working with Dates and Time</u>

<u>1. Date/Time</u>

The PHP date() function can be used to format a system date and/or a time. Also it can format a timestamp to a more readable date and time. The format is required parameter. The timestamp is optional and defaults to current system date/time.

Syntax: date(format[,timestamp])

Get a Date / Time

Here are some characters that are commonly used for date and time. Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

- d - Represents the day of the month (01 to 31)
- j – Represents the day of the month without leading zero (1 to 31)
- m - Represents a month number (01 to 12)
- n – Represents a month number without leading zero (1 to 12)
- y - Represents a year (last two digits)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week (full weekday name)
- w – Represents the day of the week (0 for Sunday, 6 for Saturday)
- F – Represents the month name (January to December)
- M – Represents the abbreviated month name (Jan to Dec)
- H - 24-hour format of an hour (00 to 23)
- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below formats today's date in three different ways:

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

The example below outputs the current time in the specified format:

```
<?php
        echo "The time is " . date("h:i:sa");
?>
```

The example below formats timestamp in three different ways:

```
<?php
        echo "Republic day - " . date("Y/m/d", 1611619200) . "<br>";
        echo "Republic day - " . date("Y.m.d", 1611619200) . "<br>";
        echo "Republic day - " . date("Y-m-d", 1611619200) . "<br>";
```

```
?>
```
Output:
```
Republic day - 2021/01/26
Republic day - 2021.01.26
Republic day - 2021-01-26
```

2.Create a Date With mktime()

The PHP mktime() function returns the timestamp for a date. This timestamp contains the number of seconds between the Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax: mktime(hour, minute, second, month, day, year)

The example below creates a date and time with the date() function from a number of parameters in the mktime() function:

e.g.:
```
<?php
        $d=mktime(0, 0, 0, 1, 26, 2021);
        echo "Republic day - " . date("Y/m/d", $d) . "<br>";
?>
```
Output:
```
Republic day - 2021/01/26
```

3.Create a Date From a String With strtotime()

The PHP strtotime() function is used to convert a human readable date string into a timestamp (the number of seconds since January 1 1970 00:00:00 GMT).

Syntax: strtotime(time, now)

The example below creates a date and time from the strtotime() function:

Example
```
<?php
        $d=strtotime("26 January 2021");
        echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```
Output:
```
Created date is 2021-01-26 12:00:00am
```

PHP is quite clever about converting a string to a date, so you can put in various values:

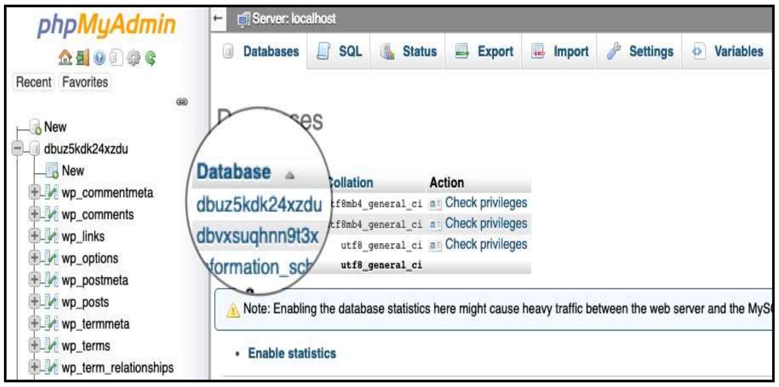
Example
```
        <?php
        $d=strtotime("tomorrow");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
        $d=strtotime("next Saturday");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
        $d=strtotime("+3 Months");
```

```
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

Creating tables using PhpMyAdmin

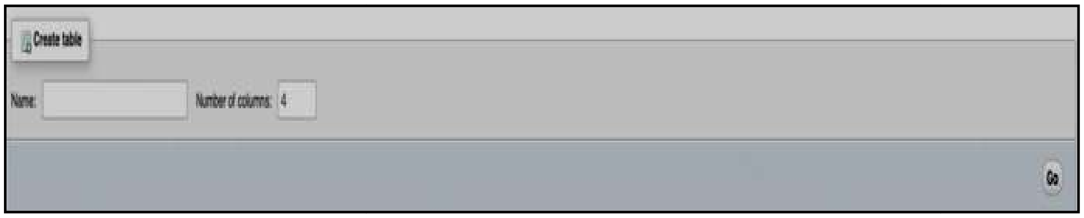
1 Creating tables using PhpMyAdmin

To create new tables inside a database, open the phpMyAdmin tool, click on the **Databases** tab and click on the name of the desired database.



Note: Enabling the database statistics here might cause heavy traffic between the web server and the MyS

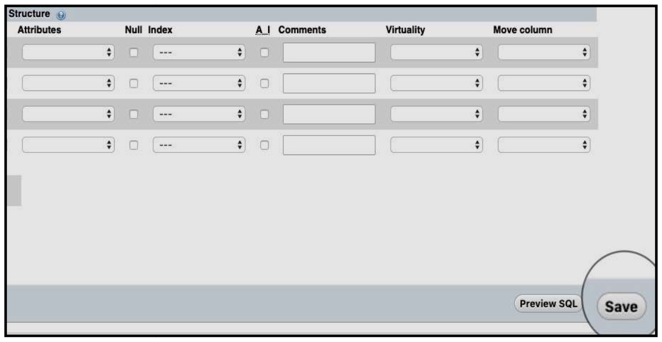
On the new page that opens you will see a list of all the current tables inside the database and a section named **Create table**. In that section, in the **Name** field, input the desired new name of the table and then select the number of columns that the table should have via the **Number of columns** drop-down. When ready, click on **Go** to create the table.



On the next page, you can configure the structure of the columns in the new table. The different fields there are:

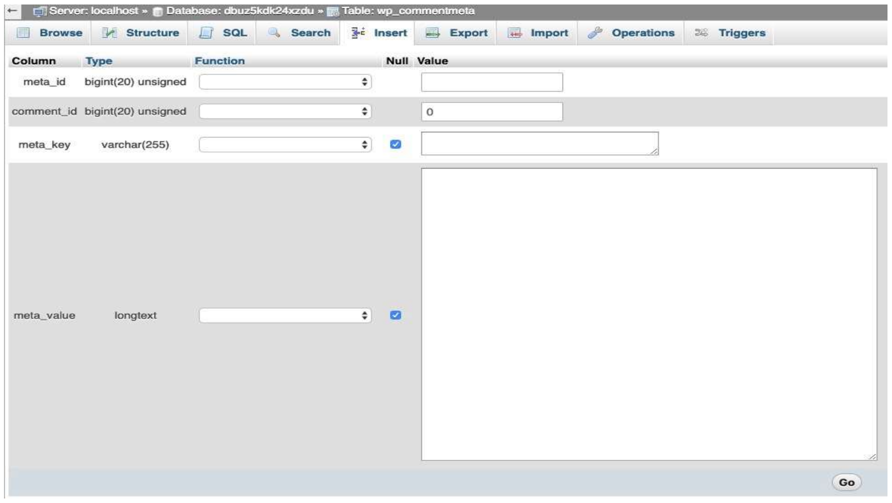
- **Name** – The name of the column;
- **Type** – The type of data, which will be stored in the corresponding column.
- **Length/Values** – The length of the field;
- **Default** – Specify if the fields in the column would have a default value. This is useful when you want to have timestamps for the entries in each row;
- **Null** – Define whether the field value can be *NULL*.
- **Index** – Set the Index of the row.
- **A_I** – Short for Auto Increment. If this option is enabled then the values in the fields of the column will be auto incremented;
- **Comments** – Here add comments, which will be included in the database SQL code.

When ready, click on **Save** to create the new table.



How to Add Content in a Database Table

To add records inside a database table, open the table with phpMyAdmin and click on the **Insert** tab.


Server: localhost » Database: dbuz5kdk24xzdu » Table: wp_commentmeta
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers
Go

Column	Type	Function	Null	Value
meta_id	bigint(20) unsigned			
comment_id	bigint(20) unsigned			0
meta_key	varchar(255)		✓	
meta_value	longtext		✓	

Enter the desired data in the corresponding fields and click on **Go** to store it. You can see the newly inserted record by clicking on the **Browse** tab.

Interaction with HTML form

1 What is Form?

When you login into a website or into your mail box, you are interacting with a form.Forms are used to get input from the user and submit it to the web server for processing.A form is an HTML tag that contains items such as input box, check boxes radio buttons etc.The form is defined using the <form>...</form> tags and controls are defined using form elements such as input.

2 When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database

3 Create a form

We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

- Opening and closing form tags <form>…</form>
- Form submission type POST or GET

- Submission URL of the PHP file that will process the submitted data
- Input fields such as input boxes, dropdowns, buttons, radio buttons, checkboxes etc.

4Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

5.1 PHP POST method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method is ideal when you do not want to display the form post values in the URL.
- A good example of using post method is when submitting login details to the server.

It has the following syntax.

```
<?php
 $_POST['variable_name'];
?>
```

5.2 PHP GET method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It’s ideal for search engine forms as it allows the users to book mark the results.

It has the following syntax.

```
<?php
$_GET['variable_name'];
?>
```

6. Processing the registration form data

When a form has been submitted, the values are populated in the super global array.We will use the PHP isset() function to check if the form values have been filled in the super global array and process the data.

Validating HTML Form

Form Validation in PHP

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and dropdown, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

1. Empty String
2. Validate String
3. Validate Numbers

4. Input length
5. Button click

1. Empty String

The code below checks that the field is not empty. If the user leaves the required field empty, it will show an error message.

```
if (empty ($_POST["name"]))
   echo"Error! You didn't enter the Name.";
else
   echo $_POST["name"];
```

2. Validate String

The code below checks that the field will contain only alphabets and whitespace, for example - name. If the name field does not receive valid input from the user, then it will show an error message. Alternatively, use a loop to step through each character.

```
$name = "TYBCA PHP";
if (!preg_match ("/^[a-zA-z]*/", $name) )
   echo "Only alphabets and whitespace are allowed.";
else
   echo $name;
```

3. Validate Number

The below code validates that the field will only contain a numeric value. **For example -** Mobile no.If the *Mobile no* field does not receive numeric data from the user, the code will display an error message:

```
$mobileno = “12345”;
if (!preg_match ("/^[0-9]{5}/", $mobileno) )
   echo "Only numeric value is allowed.";
else
   echo $mobileno;
```

4. Input Length Validation

The input length validation restricts the user to provide the value between the specified range, for Example - Mobile Number. A valid mobile number must have 10 digits.The given code will help you to apply the length validation on user input: Alternatively, use a loop to check or the above code of point 3.

```
$phone= “123456”;
$length = strlen ($phone);
if ( $length < 6 && $length > 6)
   echo "Phone must have 6 digits.";
else
   echo "Your Phone number is: " .$phone;
```

5. Button Click Validate

The codebelow validates that the user clicked on submit button and sent the form data to the server using one of the methods.

```
        if (isset($_POST['submit'])
                echo "Submit button is clicked.";
--------------------------------------------------------------------------------------
        if ($_SERVER["REQUEST_METHOD"] == "POST")
                echo "Data is sent using POST method ";
        else  if ($_SERVER["REQUEST_METHOD"] == "GET")
                echo "Data is sent using GET method ";
        else
                echo "Data is not submitted";
```

Error checking or Exiting

PHP Error Handling

Error handling in PHP is simple. An error message with filename, line number and a message describing the error is sent to the browser.When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

Different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

e.g.

```
        <?php
        $file=fopen("mytestfile.txt","r");
        ?>
```

If the file does not exist you might get an error like this:

```
        Warning: fopen(mytestfile.txt) [function.fopen]: failed to open stream:
        No such file or directory in C:\wamp\www\test.php on line 2
```

To prevent the user from getting an error message like the one above, we test whether the file exists before we try to access it:

e.g.

```
        <?php
        if(file_exists("mytestfile.txt"))
                $file = fopen("mytestfile.txt", "r");
        else
                die("Error: The file does not exist.");
        ?>
```

Or you can also write like this

```
<?php
        $file = fopen("mytestfile.txt", "r") or die("Error: The file does not exist.");
?>
```

Now if the file does not exist you get customized error like this:

```
Error: The file does not exist.
```

Introduction to Regular Expression

What is Regular Expression

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.A regular expression can be a single character, or a more complicated pattern.Regular expressions can be used to perform all types of text search and text replace operations.

Syntax

In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.

```
$pattern = "/SPU/i";
```

In the example above, **/** is the delimiter, **SPU** is the pattern that is being searched for, and **i** is a modifier that makes the search case-insensitive.

The delimiter can be any character except letter, number, backslash or space. The most common delimiter is the forward slash (**/**), but when your pattern contains forward slashes it is convenient to choose other delimiters such as **#** or **~**.

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions. The preg_match(), preg_match_all() and preg_replace() functions are some of the most commonly used ones:

Function	**Description**
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

Using preg_match()

The preg_match() function will tell you whether a string contains matches of a pattern. Alternatively use string functions to do the same.

e.g. Use a regular expression to do a case-insensitive search for "SPU" in a string:

```
<?php
        $str = "Visit SPU";
        $pattern = "/SPU/i";
        echo preg_match($pattern, $str);                // Outputs 1 indicating it is there
?>
```

Using preg_match_all()

The preg_match_all() function will tell you how many matches were found for a pattern in a string.
e.g. Use a regular expression to do a case-insensitive count of the number of occurrences of "ain" in a string:

```
<?php
        $str = "The rain in SPAIN falls mainly on the plains.";
        $pattern = "/ain/i";
        echo preg_match_all($pattern, $str);            //Outputs 4 indicating 4 occurrences
?>
```

Using preg_replace()

The preg_replace() function will replace all of the matches of the pattern in a string with another string.

e.g.

Use a case-insensitive regular expression to replace Microsoft with SPU in a string:

```
        <?php
        $str = "Visit Microsoft!";
        $pattern = "/microsoft/i";
        echo preg_replace($pattern, "SPU", $str);            // Outputs "Visit SPU!"
        ?>
```

Regular Expression Modifiers: Modifiers can change how a search is performed.

Modifier	**Description**
i	Performs a case-insensitive search

Regular Expression Patterns: Brackets are used to find a range of characters:

Expression	**Description**
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Metacharacters: Metacharacters are characters with a special meaning:

Metacharacter	**Description**
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
^	Finds a match as the beginning of a string as in: ^Hello

Quantifiers

Quantifiers define quantities:

Quantifier	**Description**
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X n*'s

Note: If your expression needs to search for one of the special characters you can use a backslash (\ ) to escape them. For example, to search for one or more question marks you can use the following expression: $pattern = '/\?+/';

Grouping

You can use parentheses ( ) to apply quantifiers to entire patterns. They also can be used to select parts of the pattern to be used as a match.

e.g.

Use grouping to search for the word "banana" by looking for ba followed by two instances of na:

```
<?php
$str = "Apples and bananas.";
$pattern = "/ba(na){2}/i";
echo preg_match($pattern, $str); // Outputs 1
?>
```

File Handling

PHP File Handling

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

PHP fopen() function is used to open file or URL and returns the resource handle. The fopen() function accepts two arguments: $filename and $mode. The $filename represents the file to be opened and $mode represents the file mode for example read-only, read-write, write-only, etc.

Syntax:resource_handle= fopen (string $filename, string $mode [,bool $use_include_path = false])

e.g.

```
<?php
        $handle = fopen("c:\\wamp\\www\\textfile.txt", "r");
?>
```

PHP Open File Mode:

Mode	Description
r	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
w	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
a	Opens file in **write-only** mode& places file pointer at end of file. If file is not found, it creates a new file.

PHP Close File - fclose()

The PHP fclose() function is used to close an open file pointer.

Syntax: **fclose (resource $handle)**

e.g.

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

Reading file in php

PHP provides various functions to read data from file. There are different functions that allow you to read all file data, read data line by line and read data character by character.

The available PHP file read functions are given below.

- fread()
- fgets()
- fgetc()

PHP Read File - fread()

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax: string = fread (resource $handle , int $length)

e.g.

```
<?php
        $filename = "c:\\wamp\\www\\testfile.txt";
        $handle = fopen($filename, "r");                    //open file in read mode
        $contents = fread($handle, filesize($filename));    //read file
        echo $contents;                                     //printing data of file
        fclose($handle);                                    //close file
?>
```

Output:

```
hellophp file
```

PHP Read File - fgets()

The PHP fgets() function is used to read a single line from file. To read multiple lines, use fgets() multiple times. To read all lines, use !feof() function inside the while loop

Syntax: string = fgets (resource $handle , int $length)

e.g.

```
<?php
        $fp = fopen("c:\\wamp\\www\\testfile.txt", "r");
        //open file in read mode
        echo fgets($fp);                                        //fetches the first line
        fclose($fp);
?>
```

PHP Read File - fgetc()

The PHP fgetc() function is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

Syntax: **string = fgetc (resource $handle)**

e.g.

```
<?php
        $fp = fopen("c:\\wamp\\www\\testfile.txt", "r");//open file in read mode
        while(!feof($fp))
                echo fgetc($fp);
        fclose($fp);
?>
```

PHP Write File - fwrite()

PHP fwrite() and fputs() functions are used to write data into file. To write data into file, you need to use **w** mode.The PHP fwrite() function is used to write content of the string into file. The return integer indicates the number of characters written in the file.

Syntax: [int =]fwrite (resource $handle , string $string [, int $length])

e.g.

```
<?php
        $fp = fopen("c:\\wamp\\www\\testfile.txt", 'w');//open file in write mode
        fwrite($fp, 'hello ');
        fwrite($fp, 'php file');
        fclose($fp);
        echo "File written successfully";
?>
```

Output:

```
File written successfully
```

Note for overwriting the file:

The above example will overwrite the content of the file. If it requires to append the content of the file just replace the "w" with "a" in file open function.

PHP Delete File - unlink()

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

Syntax :[bool =] unlink (string $filename [, resource $context])

Example

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

UNIT – 4

Topics Covered:

1. Introduction to MySQL: Features, Merits and Demerits
2. MySQL data types and constraints
3. Working with Forms PHP and MySQL Integration
4. Basic SQL Commands (Insert, Update, Delete, Select)
5. MySQL functions (mysql_connect, mysql_select_db, mysql_query, mysql_num_rows, mysql_fetch_array, mysql_fetch_field, mysql_close)
6. Generating reports using PHP and MySQL
7. Introduction and use of Session
8. Introduction and use of Cookies

<u>What is MySQL?</u>

MySQL is an open-source Relational Database Management Systembased on SQL. In real world it may be anything from a simple shopping list or the vast amounts of information in a corporate network. For the purpose of adding, accessing, and processing data stored in a computer database, you need a database management system such as MySQL Server.It is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for MySQL however, is for the purpose of a web database.The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows. Databases are useful for storing information categorically. A company may have a database for Payroll application, Purchase & Sales application, Inventory management application, etc.

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is open-source RDBMSand free software under the GNU license.

Features of MySQL

1. **Relational Database Management System (RDBMS):** MySQL is a relational database management system, based on SQL, to access and manage the records of the table.
2. **Easy to use:** MySQL is easy to use. We can build and interact with MySQL by using simple SQL.

3. **It is secure:**MySQL consists of a data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.
4. **Client/Server Architecture:**MySQL follows the client/server architecture.
5. **Free to download:**MySQL is free and can be downloaded from MySQL official site at no cost.
6. **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, we can increase this number to a theoretical limit of 8 TB of data.
7. **Speed:**MySQL is considered one of the fast databases.
8. **Compatible on many operating systems:**MySQL is compatible to run on many operating systems.
9. **Allows roll-back:**MySQL allows transactions to be rolled back and commit.
10. **High Performance:** MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture.
11. **High Productivity:**MySQL uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.
12. **Platform Independent:**It can be downloaded, installed, and executed on most of the available operating systems.
13. **GUI Support:**MySQL provides a unified visual database graphical user interface tool named "MySQL Workbench" to work with database architects, developers, and Database Administrators.
14. **Dual Password Support:**MySQL version 8.0 provides support for dual passwords: one is the current password, and another is a secondary password.

Merits of MySQL

MySQL is a free-to-use, open-source database that facilitates effective management of databases by connecting them to the software. It is a stable, reliable and powerful solution with advanced features like the following:

1. Data Security: MySQL is globally renowned for being the most secure and reliable database management system used in popular web applications like WordPress, Drupal, Joomla, Facebook and Twitter. The data security and support for transactional processing can greatly benefit any business especially if it is an eCommerce business that involves frequent money transfers.

2. On-Demand Scalability: MySQL offers unmatched scalability to facilitate the management of massive warehouses that stack terabytes of data.

3. High Performance: Whether it is an eCommerce website that receives a million queries every single day or a high-speed transactional processing system, MySQL is designed to meet even the most demanding applications while ensuring optimum speed, full-text indexes and unique memory caches for enhanced performance.

4. Round-the-clock Uptime: MySQL comes with the assurance of 24X7 uptime and offers a wide range of high availability solutions like specialized cluster servers and master/slave replication configurations.

5. **Comprehensive Transactional Support:**MySQL tops the list of robust transactional database engines available on the market. It guarantees instant deadlock identification through server-enforced referential integrity.

6. **Complete Workflow Control:**With the average download and installation time being less than 30 minutes, MySQL means usability from day one. Whether your platform is Linux, Microsoft, Macintosh or UNIX, MySQL is a comprehensive solution with self-management features that automate everything from space expansion and configuration to data design and database administration.

7. **Reduced Total Cost of Ownership:**By migrating current database apps to MySQL, enterprises are enjoying significant cost savings on new projects.

Demerits of MySQL / Disadvantages/Drawback of MySQL

Following are the few disadvantages 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 as other RDBMS like Oracle.
- 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.

MySQL Datatypes and Constraints

MySQL supports a lot number of SQL standard data types in various categories. It uses many different data types that can be categorized into: numeric, date and time, string types, spatial types, and JSON data types.

1. **Numeric Datatypes:**MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system.

Data Type	Description
TINYINT	It can be signed or unsigned. If signed, the range is from -128 to 127. If unsigned, the range is from 0 to 255. It takes 1 byte for storage.
SMALLINT	It can be signed or unsigned. If signed, the range is from -32768 to 32767. If unsigned, the range is from 0 to 65535. It requires 2 bytes for storage.
INT	It can be signed or unsigned. If signed, the range is from -2147483648 to 2147483647. If unsigned, the range is from 0 to 4294967295. It requires 4 bytes for storage.
FLOAT(m,d)	It is a floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This will default to 10,2, where 2 is the number of decimals, and 10 is the total number of digits (including decimals). It requires 2 bytes for storage.
BOOL/ BOOLEAN	It is used only for the true and false condition. It considered numeric value 1 as true and 0 as false.

2. **Date and Time Data Type:**This data type is used to represent temporal values such as date, time, datetime, timestamp, and year. When we insert the invalid value, MySQL cannot represent it, and then zero value is used.

Data Type	Maximum Size	Explanation
YEAR[(2|4)]	Year value as 2 or 4 digits.	The default is 4 digits.
DATE	Range from '1000-01-01' to '9999-12-31'.	Displayed as 'yyyy-mm-dd'.
DATETIME	Range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'yyyy-mm-ddhh:mm:ss'.

3. **String Data Types:**The string data type is used to hold plain text and binary data, for example, files, images, etc. MySQL can perform searching and comparison of string value based on the pattern matching such as LIKE operator, Regular Expressions, etc.

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	It can have a maximum size of 255 characters.	Size is the number of characters to store. Fixed-length strings. Space padded on the right to equal size characters.
VARCHAR(size)	It can have a maximum size of 255 characters.	Size is the number of characters to store. Variable-length string.
TEXT(size)	Maximum size of 65,535 characters.	Here size is the number of characters to store.

MySQL Constraints: The constraint in MySQL is used to specify the rule that allows or restricts what values/data will be stored in the table. They provide a method to ensure data accuracy and integrity. It also helps to limit the type of data that will be inserted inside the table.

Types of MySQL Constraints:

Constraints in MySQL is classified into two types:

1. **Column Level Constraints:** These constraints are applied only to a single column that limits the permitted column data.
2. **Table Level Constraints:** These constraints are applied to the entire table that limits the type of data for the whole table.

How to create constraints in MySQL: We can define the constraints during a table created by using the CREATE TABLE statement. MySQL also uses the ALTER TABLE statement to specify the constraints in the case of the existing table schema.

Constraints used in MySQL:

The following are the most common constraints used in the MySQL:

- NOT NULL
- CHECK

- DEFAULT
- PRIMARY KEY
- AUTO_INCREMENT
- UNIQUE
- INDEX
- ENUM
- FOREIGN KEY

1. **NOT NULL Constraint:** This constraint specifies that the column cannot have NULL or empty values.
2. **UNIQUE Constraint:** This constraint ensures that all values inserted into the column will be unique. It means a column cannot store duplicate values. MySQL allows us to use more than one column with UNIQUE constraint in a table.
3. **CHECK Constraint:** It controls the value in a particular column. It ensures that the inserted value in a column must be satisfied with the given condition.
4. **DEFAULT Constraint:** This constraint is used to set the default value for the particular column where we have not specified any value.
5. **PRIMARY KEY Constraint:** This constraint is used to identify each record in a table uniquely. If the column contains primary key constraints, then it cannot be null or empty. A table may have duplicate columns, but it can contain only one primary key. It always contains unique value into a column.
6. **AUTO_INCREMENT Constraint:** This constraint automatically generates a unique number whenever we insert a new record into the table. Generally, we use this constraint for the primary key field in a table.
7. **ENUM Constraint:**The ENUM data type in MySQL is a string object. It allows us to limit the value chosen from a list of permitted values in the column specification at the time of table creation. It is short for enumeration, which means that each column may have one of the specified possible values. It uses numeric indexes (1, 2, 3…) to represent string values.
8. **INDEX Constraint:**This constraint allows us to create and retrieve values from the table very quickly and easily. An index can be created using one or more than one column. It assigns a ROWID for each row in same order they were inserted into the table.
9. **Foreign Key Constraint:** This constraint is used to link two tables together. It is also known as the referencing key. A foreign key column matches the primary key field of another table. It means a foreign key field in one table refers to the primary key field of another table.

Working with Forms: PHP and MySQL Integration

A very popular way to make a web site interactive is using HTML based forms by the site. Using HTML forms, a website can interact with a visitor and:

- Obtain registration information prior to providing access to different site offerings
- Permit a visitor to submit information to the website
- Allow a visitor to upload files to the website
- Send emails and much more.

Forms are generally used for:

- User Registration
- Receiving feedback
- Online catalogs
- Request for information
- Logging in user
- Online shopping
- Surveys
- Conferencing

Introducing HTML Form Tags and Elements

- **Form Tag**
  <FORM> indicates where the form starts and </FORM> indicates where the form ends. All other form elements must be enclosed between <FORM></FORM> tags.
  Syntax:
  <form name = “formname” action = “server side php file” method = “POST/GET”>

  Name: Uniquely identifies the form.
  Action: name of the file on the web server which will process the data captured by the form, when the data captured is submitted by the Browser.
  Method: GET–transmit data using url. Size is limited to 1KB. POST–transmit data using http protocol.Size is limited to 2GB.

- **Form Elements**
  1. **Text Box**
     When placed on HTML form the textbox permits the capture of one line of text. A user will be able to key in any textual information into a Text Box.
     Syntax:
     <input type= “text” name = “textboxname” size = “” maxlength = “” value=“”>

     Input: is an HTML keyword that indicates the start of an HTML element being described.
     Type: accepts the type of a form element, in this case text.
     Name: accepts any name which is then bound to the textbox like: txt1

  2. **Text Area**
     This is a larger textbox that allows a user to key in multiple lines of text. The entered text can span across multiple lines. The size of a Text Area is specified in Rows and Cols.
     Column means number of character in a line& Rows means number of lines.
     Syntax:
     <textarea name = “textareaname” Rows = “” Cols = “”>default text</textarea>

     Name: is the name of the text area
     Rows: How many lines of text will be accepted

Cols: how many characters will fit in each row of text captured

3. **Password**

   The password field looks identical to a text box. However, when text is typed in it, the letters are hidden. This provides a level of security when keying in password into form.

   Syntax:

   <input type= “password” name = “passwordname” size = “” maxlength = “”>

   Input: is an HTML keyword that indicates the start of an HTML object being described.
   Type: accepts the type of a form element, in this case a password type textbox.
   Name: accepts any name which is then bound to the password textbox like: pwd1

4. **Radio Button (aka. Option Buttons)**

   Radio button indicate a fixed set of choices. A radio button is an empty circle with textual prompt displayed besides it. When selected, the circle is partially filled in. only one radio button from a set of radio buttons can be selected at any time. Radio buttons are grouped together by simply naming all the radio buttons used in the form with the same name.

   Syntax:

   <input type= “radio” name = “radiobuttonname” checked>

   Input: is an HTML keyword that indicates the start of an HTML object being described.
   Type: accepts the type of a form element, in this case a radio button.
   Name: accepts any name which is then bound to the radio button like:rd
   Checked: It is used to select a default radiobutton.

5. **Check Box**

   A checkbox also looks similar to a radio button but can be used for selecting multiple choices from a set. The HTML object placed on the page is a square. The square holds a tick mark in it when selected. With checkboxes, multiple choices can be made from a list of choices.

   Syntax:

   <input type= “checkbox” name = “checkboxname” checked>

   Input: is an HTML keyword that indicates the start of an HTML object being described.
   Type: accepts the type of a form element, in this case a checkbox.
   Name: accepts any name which is then bound to the checkbox like: chk1
   Checked: Used to show any checkbox as checked by default.

6. **Drop Down Box or List Box**

   The combo box or drop down list box displays multiple options available. The combo box or drop down list box initially displays one item. When clicked on, it drops down to display a list of item to select from.

   Syntax:

```
<select name = “selectboxname” size = “1”>
<option value=””> opt1 </option>
<option value=”” selected> opt2 </option>
<option value=””> opt3 </option>
</select>
```

Select: indicates the start of the combo box.
Name: accepts any name which is then bound to the combo box like:sel
Size: accepts the size of a box. Here, 1 means one line is shown by default. A value more than 1 converts the drop down control into a listbox control.
Option: accepts a string that will be displayed when the combo box is clicked and drops down.
Value: Assigns a value to the options.
Selected: Makes an option selected by default.

7. **Submit Button**
After placing all the form elements required on an HTML page within the <form></form> tags, there should be some way to have the contents of these elements submitted/sent to a particular destination. Submit button sends the request and form data back to web server.
Syntax:
```
<input type= “submit” name = “submitbuttonname” value = “caption”>
```

Input: is an HTML keyword that indicates the start of an HTML object being described.
Type: accepts the type of a form element, in this case a submit button.
Name: accepts any name which is then bound to the submit button like: btn1
Value: accepts a string that will display on a submit button.

8. **Reset Button**
Some forms also allow erasing all the entries made in form elements prior to filling the form all over again. This functionalityresets all form elements to their default values.
Syntax:
```
<input type= “reset” name = “resetbuttonname” value = “caption”>
```

Input: is an HTML keyword that indicates the start of an HTML object being described.
Type: accepts the type of a form element, in this case a reset button.
Name: accepts any name which is then bound to the reset button like:rst1
Value: accepts a string that will display on a reset button.

Basic SQL Commands

1. **Insert:**
Insert statement is used to insert a new row in a database table.
Syntax:

It is possible to write the insert into statement in two forms.

1) The first does not specify the column names where the data will be inserted, only their values. Must be used only when values are in the same order as the column order in the table.

   Insert into table_name values (value1, value2, value3, …)

2) The second form specifies both the column names and the values to be inserted.

   Insert into table_name (column1, column2, coumn3, …) values (value1, value2, value3,…)

For e.g.:
insert into student values('1', "XYZ", "TYITM", "SEMCOM")

2. **Update:**

The update statement is used to update existing record(s) in a database table.

Syntax:

updatetable_name set column1=value , column2=value,… where some_column=some_value

where clause specifies which record or records that should be updated. If you omit the where clause, all records will be updated !

For e.g.:
update student set name="ABC", class="TYBCA", college="SEMCOM" where id= '1')

3. **Delete:**

The delete statement is used to delete existing record(s) from a database table.

Syntax:

Delete from table_name where some_column=some_value

where clause specifies which record or records to be deleted. If you omit thewhere clause, all records will be deleted !

For e.g.:
delete from student where id= '1'

4. **Select:**

The select statement is used to select and fetch the records from a database table.

Syntax:

Select */column_names from table_namewhere some_column=some_value

whereclause specifies which record or records that should be selected orfetched. If you omit the where clause, all records will be selected or fetched!

For e.g.:
select * from student

Database functions: PHP MySQL connect to a Database.The free MySQL database is very often used with PHP.

1. **Create a connection to a MySQL Database**

   Before you can access data in a database, you must create a connection to the database. This is done with the mysqli_connect() function.

   Syntax:

   mysqli_connect (servername, username, password, databasename)

   Servername: specifies server to connect to. (Default is localhost)
   Username: specifies the username to log in with. (Default is root)
   Password: specifies the password to log in with. (Default is “”- NULL)
   Databasename: specifies the name of the database to which we want toconnect.

   Example:

```
<?php
    $con = mysqli_connect(“localhost”, “root”, “”, “tybca”);
    if (!$con)
        echo “Error connecting the database !!!”;
    else
    {
        // code to execute on successful connection
    }
?>
```

2. **Closing a connection**

   The connection will be closed automatically when the script ends. To close the connection prematurely, use the mysqli_close() function.

   Syntax:

   mysqli_close(connection)

   Example:

```
<?php
    $con = mysqli_connect(“localhost”, “root”, “”, “tybca”);
    if (!$con)
        echo “Error connecting the database !!!”;
else
    {
    // code to execute on successful connection
    mysqli_close ($con);
    }
?>
```

3. **Executing a query**

   The mysqli_query() function executes a query on a MySQL database.

This function returns the query handle for SELECT and TRUE/FALSE for other queries, or FALSE on failure.

Syntax:

mysqli_query (connection,query)

query: specifies the SQL query to send

connection: specifies MySQL connection

Example:

```
<?php
    $con = mysqli_connect("localhost", "root", "", "tybca");
    if (!$con)
        echo "Error connecting the database !!!";
    else
    {
        $ins = "insert into student values ('1', 'xyz', 'tybca', 'SPU')";
        mysqli_query ($con, $ins);
        mysqli_close ($con);
    }
?>
```

4. **Getting number of rows**

The mysqli_num_rows() function returns the number of rows in a recordset retrieved using SELECT query. This function returns FALSE on failure.

Syntax:

mysqli_num_rows (data)

data: specifies which data pointer to use. The data pointer is the resultfrom the mysqli_query() function.

Example:

```
<?php
    $con = mysqli_connect("localhost", "root", "", "tybca");
    if (!$con)
            echo "Error connecting the database !!!";
    else
    {
            $sel = "select * from student";
            $result=mysqli_query ($con, $sel);
            echomysqli_num_rows ($result);
            mysqli_close ($con);
    }
?>
```

If we have 4 rows in student table then we will have following output.

Output

```
4
```

5. **Fetching Rows**

   The mysqli_fetch_row() function returns a row from a recordset when using the SELECT query. This function gets a row from the mysqli_query() function and returns a row on success, or FALSE on failure or when there are no more rows.

   Syntax:

   mysqli_fetch_row(data)

   data: specifies which data pointer to use. The data pointer is the result from the mysqli_query() function.

   Example:

```
<?php
    $con = mysqli_connect("localhost", "root", "", "tybca");
    if (!$con)
            echo "Error connecting the database !!!";
    else
    {
            $sel = "select * from student";
            $result=mysqli_query ($con, $sel);
            while ($row=mysqli_fetch_row($result))
            {
                    echo $row[0]. "<br>";
                    echo $row[1]. "<br>";
                    echo $row[2]. "<br>";
                    echo $row[3]. "<br>";
            }
        mysqli_close ($con);
    }
?>
```

   If we have 1 row in student table then we will have following output.

   Output

```
1 XYZ   TYBCASPU
```

6. **Selecting a specific database at later stage**

   To select a specific database at a later stage, mysqli_select_db() function can be used.

   Syntax:

   mysqli_select_db(connection,databasename)

connection: specifies MySQL connection
databasename: specifies the name of the database to which we want to connect the script.

Example:

```
<?php
    $con = mysqli_connect(“localhost”, “root”, “”)
    if (!$con)
            echo “Error connecting the database !!!”;
    else
    {
            mysqli_select_db($con, “tybca”);
            // code to execute on successful connection
            mysqli_close($con);
    }
?>
```

7. mysqli_fetch_array()

The mysqli_fetch_array() function returns a row from a recordset as an associative array and/or a numeric array.This function gets a row from the mysqli_query() function and returns an array on success, or FALSE on failure or when there are no more rows.

Syntax

mysqli_fetch_array(data,array_type)

Parameter	**Description**
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysqli_query() function
array_type	Optional. Specifies what kind of array to return. Possible values: • MYSQL_ASSOC - Associative array • MYSQL_NUM - Numeric array • MYSQL_BOTH - Default. Both associative and numeric array

Example

```
<?php
$con = mysqli_connect("localhost", "root", "",”TYBCA”);
if (!$con)
        die('Could not connect: ' . mysqli_error());
$sql = "SELECT * from student WHERE class='TYBCA'";
$result = mysqli_query($con ,$sql);
print_r(mysqli_fetch_array($result));
mysqli_close($con);
?>
```

8. mysqli_fetch_field()

The mysqli_fetch_field() function returns an object containing information of a field from a recordset.

This function gets field data from the mysqli_query() function and returns an object on success, or FALSE on failure or when there are no more rows.

Possible return values:

- name - Field name
- table - The table the field belongs to
- def - Default value for the field
- max_length– Maximum width of field
- type - Field type
- length – width of field as specified in table definition
- decimals – for integer fields specifies the number of decimals used

Syntax

mysqli_fetch_field(data,field_offset)

Parameter	**Description**
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysqli_query() function
field_offset	Optional. Specifies which field to start returning. 0 indicates the first field. If this parameter is not set, it will retrieve the next field

Example

```
<?php
        $con = mysqli_connect("localhost", "root", "", "feb02");
        $sql = "SELECT * from first_table";
        $result = mysqli_query($con,$sql);
        while ($property = mysqli_fetch_field($result))
        {
                echo "Field name: " . $property->name . "<br />";
                echo "Table name: " . $property->table . "<br />";
                echo "Default value: " . $property->def . "<br />";
                echo "Max length: " . $property->max_length . "<br />";
                echo "length: " . $property->length . "<br />";
                echo "type: " . $property->type . "<br />";
                echo "decimal: " . $property->decimals . "<br />";
        }
        mysqli_close($con);
?>
```

Report Generation with PHP and MySQL

MySQL database stores all the data which can be retrieved using PHP and the data can be displayed in a format required. The data can be fetched and displayed in a tabular fashion.

Example:

```
<?php
        $con = mysqli_connect("localhost", "root", "", "feb02");
        if (!$con)
                echo "Error connecting the database !!!";
        else
        {
                $sel = "select * from first_table";
                $result=mysqli_query ($con, $sel);
                echo "<table border='1'>";
                echo "<tr>";
                echo "<td>Roll Number</td>";
                echo "<td>Student Name</td>";
                echo "<td>Class</td>";
                echo "<td>Gender</td>";
                echo "</tr>";
                while ($row=mysqli_fetch_row($result))
                {
                        echo "<tr>";
                        echo "<td>$row[0]</td>";
                        echo "<td>$row[1]</td>";
                        echo "<td>$row[2]</td>";
                        echo "<td>$row[3]</td>";
                        echo "</tr>";
                }
                echo "</table>";
                mysqli_close ($con);
        }
?>
```

Introduction to Cookie:

What is a Cookie

A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visits the same website next time. You should not store sensitive information in a cookie since this value is stored on the user's computer.

Setting a Cookie in PHP

The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set.

The basic syntax of this function can be given with:
setcookie(name, value, expire, path, domain, secure)

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

Parameter	Description
name	The name of the cookie.
value	The value of the cookie.
expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
path	Optional. Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.
domain	Optional. Specify the domain for which the cookie is available to e.g www.abc.com.
secure	Optional. This field indicates that cookie should be sent only if a secure HTTPS connection exists.

Here's an example that uses setcookie() function to create a cookie named username and assign the value John Carter to it. It also specify that the cookie will expire after 30 days = (today+30d*24hr*60min*60sec).

Example

```
<?php
	setcookie("username", "John Carter", time()+30*24*60*60);	// Setting a cookie
?>
```

Accessing Cookies Values

The PHP $_COOKIE superglobal variable is used to retrieve a cookie value. It typically is an associative array that contains a list of all the cookie values sent by the browser in the current request accessible by cookie name. The individual cookie value can be accessed using standard array notation, for example to display the username cookie set in the previous example, you could use the following code.

Example

```
<?php
	echo $_COOKIE["username"];			// Accessing an individual cookie value
?>
```

The PHP code in the example above will produce the following output.

```
John Carter
```

It's a good practice to check whether a cookie is set or not before accessing its value. To do this you can use the PHP isset() function, like this:

Example

```
<?php
        if(isset($_COOKIE["username"]))                // Verifying whether a cookie is set or not
                echo "Hi " . $_COOKIE["username"];
        else
                echo "Welcome Guest!";
?>
```

You can use the print_r() function like print_r($_COOKIE); to see the structure of $_COOKIE associative array, like you with other arrays.

Removing Cookies

You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need to set the expiration date to a past Timestamp value, as shown in the example below:

Example

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

Introduction to Session:

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer.Although you can store data using cookies but it has some security issues. Since cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application.Also every time the browser requests a URL to the server, all the cookie data for that website is automatically sent to the server along with the request. It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance.You can solve both of these issues by using the PHP session.

A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called session identifier or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Why Session is used?

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.

How session works?

PHP, a session provides a way to store web page visitor preferences on a web server in the form of variables that can be used across multiple pages. The information is retrieved from the web server when a session is opened at the beginning of each web page. The session expires when the web page is closed.

Starting a PHP Session

To begin a new session, simply call the PHP **session_start()** function. It will create a new session and generate a unique session ID for the user. The PHP code in the example below simply starts a new session.

The PHP code in the example below simply starts a new session.

Example

```
<?php
        session_start();                // Starting session
?>
```

The session_start() function first checks to see if a session already exists by looking for the presence of a session ID. If it finds one, i.e. if the session is already started, it sets up the session variables and if session doesn't exist, it starts a new session by creating a new session ID.

Storing and Accessing Session Data

You can store all your session data as key-value pair in the $_SESSION[] superglobal array. The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

Example

```
<?php
        session_start();                // Starting session
        $_SESSION["name"] = "Peter";                // Stores session data
?>
```

To access the session data from any other page on the same web domain, simply recreate the session by calling session_start() and then pass the corresponding key to
the $_SESSION associative array.

Example

```
<?php
        session_start();                // Starting session
        echo“Hello “ . $_SESSION["name"];           // Accessing session data
?>
```

The PHP code in the example above will produce the following output.

```
Hi, Peter Parker
```

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array, as shown in the following example:

Example

```
<?php
        session_start();                // Starting session
        if(isset($_SESSION["lastname"]))               // Removing session data
                unset($_SESSION["lastname"]);
?>
```

However, to destroy a session completely, simply call the session_destroy() function. This function does not need any argument and a single call destroys all the session data.

Example

```
<?php
        session_start();                // Starting session
        session_destroy();              // Destroying session
?>
```

Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity.