INSTITUTE OF MANAGEMENT AND INFORMATION TECHNOLOGY

Subject-PHP & MYSQL(MCA 406E)

RACHANA BEHERA

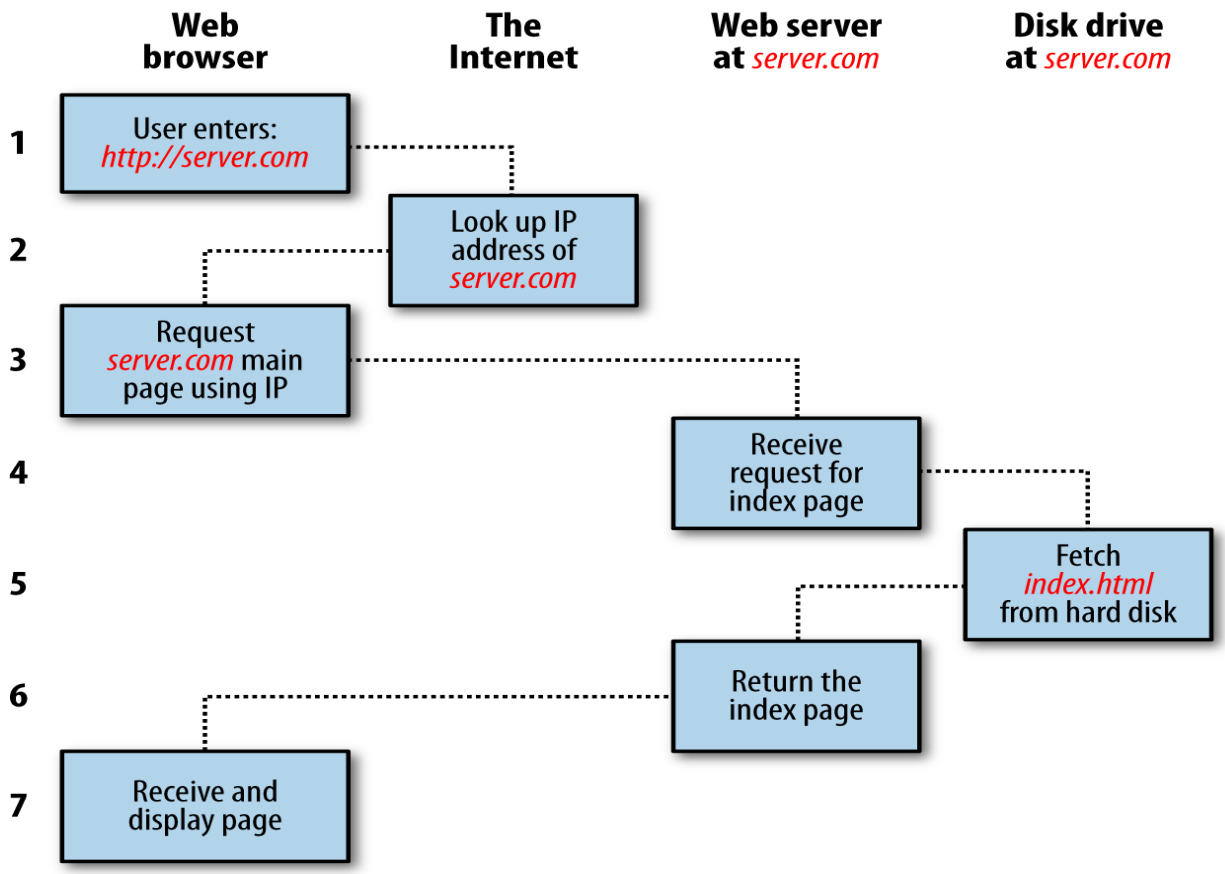
Email-id: rachanabehera2792@gmail.com

UNIT 1

INTRODUCTION TO PHP:

❖ Evaluation of PHP:

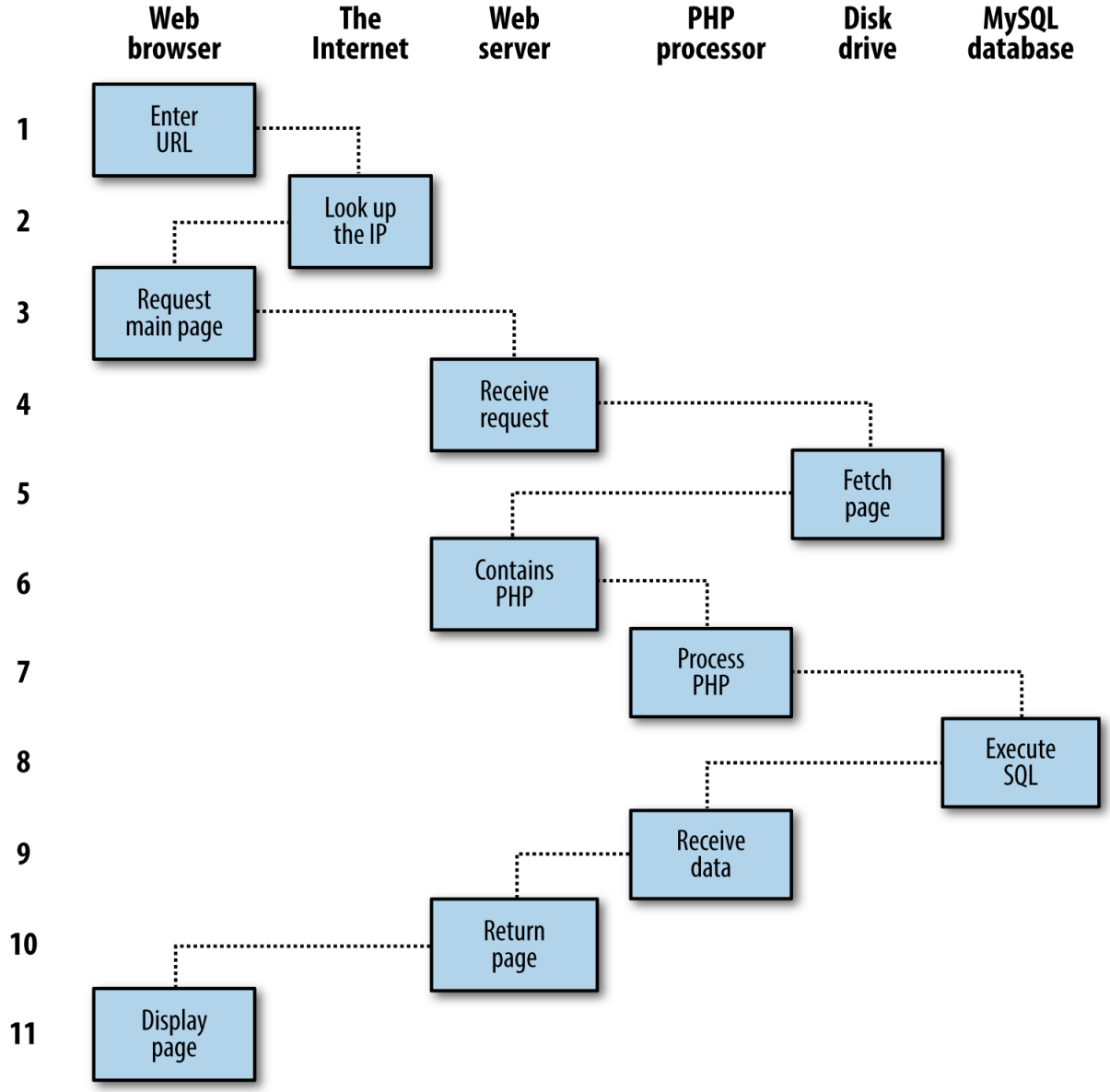
- The combination of PHP and MySQL is the most convenient approach to dynamic, database-driven web design, holding its own in the face of challenges from integrated frameworks—such as Ruby on Rails—that are harder to learn. Due to its open source roots (unlike the competing Microsoft .NET Framework), it is free to implement and is therefore an extremely popular option for web development.
- At its most basic level, the request/response process consists of a web browser asking the web server to send it a web page and the server sending back the page. The browser then takes care of displaying the page.
-



(The basic client/server request/response sequence)

- Each step in the request and response sequence is as follows:
  1. You enter *http://server.com* into your browser's address bar.
  2. Your browser looks up the IP address for *server.com*.
  3. Your browser issues a request for the home page at *server.com*.
  4. The request crosses the Internet and arrives at the *server.com* web server.
  5. The web server, having received the request, looks for the web page on its disk.
  6. The web page is retrieved by the server and returned to the browser.
  7. Your browser displays the web page.

- For dynamic web pages, the procedure is a little more involved, because it may bring both PHP and MySQL into the mix.



1. You enter *http://server.com* into your browser's address bar.
2. Your browser looks up the IP address for *server.com*.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the *server.com* web server.
5. The web server, having received the request, fetches the home page from its hard disk.
6. With the home page now in memory, the web server notices that it is a file incorporating PHP scripting and passes the page to the PHP interpreter.
7. The PHP interpreter executes the PHP code.
8. Some of the PHP contains MySQL statements, which the PHP interpreter now passes to the MySQL database engine.
9. The MySQL database returns the results of the statements to the PHP interpreter.
10. The PHP interpreter returns the results of the executed PHP code, along with the results from the MySQL database, to the web server.
11. The web server returns the page to the requesting client, which displays it.

➢ PHP:

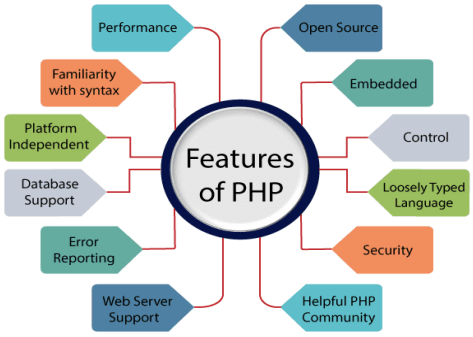
- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).
- PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995. **PHP 7.4.0**is the latest version of PHP, which was released on **28 November**. Some important points need to be noticed about PHP are as followed:
  - ✓ PHP stands for Hypertext Preprocessor.
  - ✓ PHP is an interpreted language, i.e., there is no need for compilation.
  - ✓ PHP is faster than other scripting languages, for example, ASP and JSP.
  - ✓ PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
  - ✓ PHP can be embedded into HTML.
  - ✓ PHP is an object-oriented language.
  - ✓ PHP is an open-source scripting language.
  - ✓ PHP is simple and easy to learn language.

➢ Why use PHP:

- PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.
- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. **For example** - Registration form.

PHP Features:

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



- **Performance:** PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.
- **Open Source:** PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.
- **Familiarity with syntax:** PHP has easily understandable syntax. Programmers are comfortable coding with it.
- **Embedded:** PHP code can be easily embedded within HTML tags and script.
- **Platform Independent:** PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- **Database Support:** PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
- **Error Reporting** -PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.
- **Loosely Typed Language:** PHP allows us to use a variable without declaring its data type. It will be taken automatically at the time of execution based on the type of data it contains on its value.
- **Web servers Support:** PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
- **Security:** PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
- **Control:** Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

- By default, PHP documents end with the extension *.php*. When a web server encounters this extension in a requested file, it automatically passes it to the PHP processor. To trigger the PHP commands, you need to learn a new tag. Here is the first part:
  <?php
  The first thing you may notice is that the tag has not been closed. This is because entire sections of PHP can be placed inside this tag, and they finish only when the closing part is encountered, which looks like this:
  ?>
  A small PHP "Hello World" program might look like:
  Example: Invoking PHP

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

The Structure of PHP:

- Using Comments:
  There are two ways in which you can add comments to your PHP code. The first turns a single line into a comment by preceding it with a pair of forward slashes:

```
// This is a comment
```

  This version of the comment feature is a great way to temporarily remove a line of code from a program that is giving you errors. For example, you could use such a comment to hide a debugging line of code until you need it, like this:

```
// echo "X equals $x";
```

  You can also use this type of comment directly after a line of code to describe its action, like this:

```
$x += 10; // Increment $x by 10
```

  When you need multiple-line comments, there's a second type of comment, which looks like:
  Example: *A multiline comment*

```
<?php
/* This is a section
of multiline comments
which will not be
interpreted */
?>
```

  You can use the /* and */ pairs of characters to open and close comments almost anywhere you like inside your code. Most, if not all, programmers use this construct to temporarily comment out entire sections of code that do not work or that, for one reason or another, they do not wish to be interpreted.

❖ Basic Syntax:

- PHP is quite a simple language with roots in C and Perl, yet it looks more like Java. It is also very flexible, but there are a few rules that you need to learn about its syntax and structure.

➢ Semicolons:

You may have noticed in the previous examples that the PHP commands ended with a semicolon, like this:

```
$x += 10;
```

Probably the most common cause of errors you will encounter with PHP is forgetting this semicolon. This causes PHP to treat multiple statements like one statement, which it is unable to understand, prompting it to produce a Parse error message.

➢ The $ symbol:

- The $ symbol has come to be used in many different ways by different programming languages.
- For example, if you have ever written in the BASIC language, you will have used the $ to terminate variable names to denote them as strings.
- In PHP, however, you must place a $ in front of *all* variables. This is required to make the PHP parser faster, as it instantly knows whenever it comes across a variable.
- Whether your variables are numbers, strings, or arrays, they should all look something like those in Example.

  Example: Three different types of variable assignment

```
<?php
$mycounter = 1;
$mystring = "Hello";
$myarray = array("One", "Two", "Three");
?>
```

- PHP leaves you completely free to use (or not use) all the indenting and spacing you like. In fact, sensible use of *whitespace* is generally encouraged (along with comprehensive commenting) to help you understand your code when you come back to it. It also helps other programmers when they have to maintain your code.

❖ Defining Variables and Constants:

➢ Variable:

- Variable-naming rules:

  When creating PHP variables, you must follow these four rules:

  1. Variable names must start with a letter of the alphabet or the _ (underscore) character.
  2. Variable names can contain only the characters a-z, A-Z, 0-9, and _ (underscore).
  3. Variable names may not contain spaces. If a variable must comprise more than one word, it should be separated with the _ **(**underscore**)** character (e.g., $user_name).
  4. Variable names are case-sensitive. The variable $High_Score is not the same as the variable $high_score.

- To allow extended ASCII characters that include accents, PHP also supports the bytes from 127 through 255 in variable names. But unless your code will be maintained only by programmers who are used to those characters, it's probably best to avoid them, because programmers using English keyboards will have difficulty accessing them.

- ## String variables:

Assigning a string value to a variable, like this:

```
$username = "Fred Smith";
```

The quotation marks indicate that “Fred Smith” is a *string* of characters. You must enclose each string in either quotation marks or apostrophes (single quotes).

```
echo $username;
```

Or you can assign it to another variable (photocopy the paper and place the copy inanother matchbox), like this:

```
$current_user = $username;
```

- ## Example:

```
<?php //
$username = "Fred Smith";
echo $username;
echo "<br>";
$current_user = $username;
echo $current_user;
?>
```

- ## Numeric variables:

Variables don’t contain just strings—they can contain numbers too.

```
$count = 17;
```

You could also use a floating-point number (containing a decimal point); the syntax is the same:

```
$count = 17.5;
```

In PHP, you would assign the value of $count to another variable or perhaps just echo it to the web browser.

- ## Arrays:

The equivalent of this in PHP would be the following:

```
$team = array ('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```

The array building code consists of the following construct:

```
array ();
```

with five strings inside. Each string is enclosed in apostrophes. If we then wanted to know who player 4 is, we could use this command:

```
echo $team[3]; // Displays the name Chris
```

The reason the previous statement has the number 3, not 4, is because the first element of a PHP array is actually the zeroth element, so the player numbers will therefore be 0 through 4.

- ## PHP $ and $$ Variables:

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

- Example :

```
1. <?php
2. $x = "abc";
3. $$x = 200;
4. echo $x."<br/>";
5. echo $$x."<br/>";
6. echo $abc;
7. ?>
```

Output:

abc
200
200

- In the above example, we have assigned a value to the variable **x** as **abc**. Value of reference variable **$$x** is assigned as **200**.
- Now we have printed the values **$x, $$x** and **$abc**.

Example:

```
1. <?php
2.  $x="U.P";
3. $$x="Lucknow";
4. echo $x. "<br>";
5. echo $$x. "<br>";
6. echo "Capital of $x is " . $$x;
7. ?>
```

Output:

U.P
Lucknow
Capital of U.P is Lucknow**.**

- In the above example, we have assigned a value to the variable **x** as **U.P**. Value of reference variable **$$x** is assigned as **Lucknow.**

- Now we have printed the values **$x, $$x** and a string.
  Example:

```
1. <?php                                    Output:
2. $name="Cat";                             Cat
3. ${$name}="Dog";                          Dog
4. ${${$name}}="Monkey";                    Dog
5. echo $name. "<br>";                      Monkey
6. echo ${$name}. "<br>";                   Monkey
7. echo $Cat. "<br>";
8. echo ${${$name}}. "<br>";
9. echo $Dog. "<br>";
10. ?>
```

- In the above example, we have assigned a value to the variable name **Cat**. Value of reference variable **${$name}** is assigned as **Dog** and **${${$name}}** as **Monkey**.
- Now we have printed the values as **$name, ${$name}, $Cat, ${${$name}}** and **$Dog.**

PHP Variable Scope:

The scope of a variable is defined as its range in the program under which it can be accessed. In other words, "The scope of a variable is the portion of the program within which it is defined and can be accessed."

PHP has three types of variable scopes:

Local variable:

- ✓ The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.
- ✓ A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example:

✓ ***File: local_variable1.php***

```
1. <?php
2.     function local_var()
3.     {
4.         $num = 45;  //local variable
5.         echo "Local variable declared inside the function is: ". $num;
6.     }
7.     local_var();
8. ?>
```

Output:

Local variable declared inside the function is: 45

- Global variable:

✓ The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program.

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

✓ Let's understand the global variables with the help of an example:

✓ Example: *File: global_variable1.php*

```
1. <?php
2.     $name = "Sanaya Sharma";       //Global Variable
3.     function global_var()
4.     {
5.         global $name;
6.         echo "Variable inside the function: ". $name;
7.         echo "</br>";
8.     }
9.     global_var();
10.    echo "Variable outside the function: ". $name;
11. ?>
```

Output:

Variable inside the function: Sanaya Sharma
Variable outside the function: Sanaya Sharma

- Static variable:
  - ✓ It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.
  - ✓ Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope. Understand it with the help of an example:
  - ✓ Example: *File: static_variable.php*

```
1. <?php
2.     function static_var()
3.     {
4.         static $num1 = 3;       //static variable
5.         $num2 = 6;          //Non-static variable
6.         //increment in non-static variable
7.         $num1++;
8.         //increment in static variable
9.         $num2++;
10.        echo "Static: " .$num1 ."</br>";
11.        echo "Non-static: " .$num2 ."</br>";
12.    }
13.
14. //first function call
15.    static_var();
16.
17.    //second function call
18.    static_var();
19. ?>
```

Output:

```
Static: 4
Non-static: 7
Static: 5
Non-static: 7
```

➢ PHP Constants:

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

1. Using define() function
2. Using const keyword

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

- PHP constant: define()

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

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

1. **name:** It specifies the constant name.
2. **value:** It specifies the constant value.
3. **case-insensitive:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

✓ Let's see the example to define PHP constant using define().
File: constant1.php

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP");
3. echo MESSAGE;
4. ?>
```

Output:

```
Hello JavaTpoint PHP
```

✓ Create a constant with **case-insensitive** name:
File: constant2.php

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
3. echo MESSAGE, "</br>";
4. echo message;
5. ?>
```

Output:

```
Hello JavaTpoint PHP
Hello JavaTpoint PHP
```

- ## PHP constant: const keyword

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

File: constant4.php

```
<?php
const MESSAGE="Hello const by JavaTpoint PHP";
echo MESSAGE;
?>
```

Output:

```
Hello const by JavaTpoint PHP
```

- ## Constant() function:

There is another way to print the value of constants using constant() function instead of using the echo statement.

Syntax: constant (name)

File: constant5.php

```
<?php
    define("MSG", "JavaTpoint");
    echo MSG, "</br>";
    echo constant("MSG");
    //both are similar
?>
```

Output:

```
JavaTpoint
JavaTpoint
```

- Constant vs Variables

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

❖ PHP Data Types:

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

➢ Scalar Types:

It holds only single value. There are 4 scalar data types in PHP.

1. boolean

   Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**. It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

   Example:

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

   Output:

```
This condition is TRUE.
```

2. Integer

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

   Rules for integer:

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

   Example:

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

Output:

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

3. Float

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

Example:

```
<?php
    $n1 = 19.34;
    $n2 = 54.472;
    $sum = $n1 + $n2;
    echo "Addition of floating numbers: " .$sum;
?>
```

Output:

```
Addition of floating numbers: 73.812
```

4. String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within **single quotes** or in **double quotes**. But both are treated differently. To clarify this, see the example below:

Example:

```
<?php
    $company = "Javatpoint";
    //both single and double quote statements will treat different
    echo "Hello $company";
    echo "</br>";
    echo 'Hello $company';
?>
```

Output:

```
Hello Javatpoint
Hello $company
```

Compound Types:

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

1. array

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

   Example:

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

   Output:

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

2. object

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

   Example:

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

   Output:

```
Bike Model: Royal Enfield
```

- ## Special Types:

There are 2 special data types in PHP.

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

2. NULL
   Null is a special data type that has only one value: **NULL**. There is a convention of writing it in capital letters as it is case sensitive.
   The special type of data type NULL defined a variable with no value.
   Example:

```
1. <?php
2.     $nl = NULL;
3.     echo $nl;   //it will not give any output
4. ?>
```

Output:

❖ OPERATOR AND EXPRESSION:

- PHP Operator is a symbol i.e. used to perform operations on operands. In simple words, operators are used to perform operations on variables or values.

- An expression is a combination of values, variables, operators, and functions that results in a value. It's familiar to anyone who has taken high-school algebra:

  $y = 3(\text{abs}(2x) + 4)$

  which in PHP would be

  $y = 3 * (abs(2 * $x) + 4);

- For example:
  $num=10+20;//+ is the operator and 10,20 are operands
  In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

- PHP Operators can be categorized in following forms:
- Arithmetic Operators:
- The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

- Assignment Operators:

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

Bitwise Operators:

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

Comparison Operators:

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b

<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

- Logical Operators:

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

Incrementing/Decrementing Operators:

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

String Operators:

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Array Operators:

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order

!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

- Type Operators:

The type operator **instanceof** is used to determine whether an object, its parent and its derived class are the same type or not. Basically, this operator determines which certain class the object belongs to. It is used in object-oriented programming.

```
<?php
    //class declaration
    class Developer
    {}
    class Programmer
    {}
    //creating an object of type Developer
    $charu = new Developer();
    //testing the type of object
    if( $charu instanceof Developer)
    {
        echo "Charu is a developer.";
    }
    else
    {
        echo "Charu is a programmer.";
    }
    echo "</br>";
    var_dump($charu instanceof Developer);          //It will return true.
    var_dump($charu instanceof Programmer);       //It will return false.
?>
```

Output:

```
Charu is a developer.
bool(true) bool(false)
```

- Execution Operators:
- PHP has an execution operator **backticks (``)**. PHP executes the content of backticks as a shell command. Execution operator and **shell_exec()** give the same result.

Operator	Name	Example	Explanation
``	backticks	echo `dir`;	Execute the shell command and return the result. Here, it will show the directories available in current folder.

- Error Control Operators:

PHP has one error control operator, i.e., **at (@) symbol**. Whenever it is used with an expression, any error message will be ignored that might be generated by that expression.

Operator	Name	Example	Explanation
@	at	@file ('non_existent_file')	Intentional file error

- We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

Unary Operators: works on single operands such as ++, -- etc.

Binary Operators: works on two operands such as binary +, -, *, / etc.

Ternary Operators: works on three operands such as "?:".

DECISIONS AND LOOP:

❖ Making Decisions:

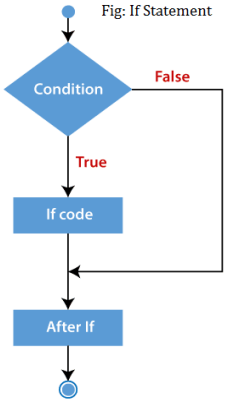
- PHP conditional statements allow you to make a decision, based upon the result of a condition. These statements are called as Decision Making Statements or Conditional Statements.
- There are various ways to use if statement in PHP.

1. If Statement:

✓ PHP if statement allows conditional execution of code. It is executed if condition is true. If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

Syntax

```
1. if(condition){
2. //code to be executed
3. }
```



✓ **Example:**

```
1. <?php
2. $num=12;
3. if($num<100){
4. echo "$num is less than 100";
5. }
6. ?>
```

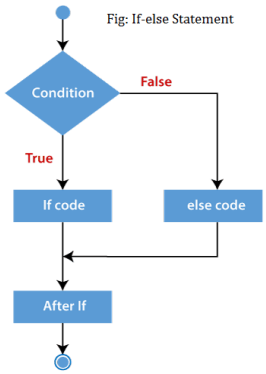
Output:

12 is less than 100

2. If-else Statement:

✓ PHP if-else statement is executed whether condition is true or false. If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.

✓ **Syntax**




```
1. if(condition){
2. //code to be executed if true
3. }
4. else{
5. //code to be executed if false
6. }
```

✓ **Example**

```
1. <?php
2. $num=12;
3. if($num%2==0){
4. echo "$num is even number";
5. }else{
6. echo "$num is odd number";
7. }
8. ?>
```

Output:

12 is even number

3. If-else-if Statement:

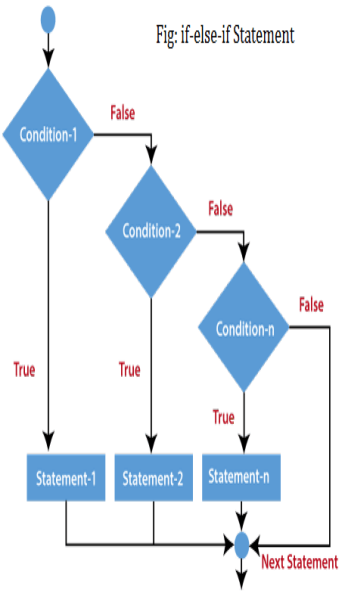


Fig: if-else-if Statement

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

```
1. if (condition1){
2. //code to be executed if condition1 is true
3. } elseif (condition2){
4. //code to be executed if condition2 is true
5. } elseif (condition3){
6. //code to be executed if condition3 is true
7. ....
8. } else{
9. //code to be executed if all given conditions are false
10. }
```

- ✓ **Example**

```
1. <?php
2.     $marks=69;
3.     if ($marks<33){
4.         echo "fail";
5.     }
6.     else if ($marks>=34 && $marks<50) {
7.         echo "D grade";
8.     }
9.     else if ($marks>=50 && $marks<65) {
10.        echo "C grade";
11.    }
12.    else if ($marks>=65 && $marks<80) {
13.        echo "B grade";
14.    }
15.    else if ($marks>=80 && $marks<90) {
16.        echo "A grade";
```

```
17.     }
18.     else if ($marks>=90 && $marks<100) {
19.         echo "A+ grade";
20.     }
21.     else {
22.         echo "Invalid input";
23.     }
24. ?>
```

Output:

B Grade

4. PHP nested if Statement:

✓ The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is **true**.

✓ **Syntax**

```
1. if (condition) {
2. //code to be executed if conditio
   n is true
3. if (condition) {
4. //code to be executed if conditio
   n is true
5. }
6. }
```

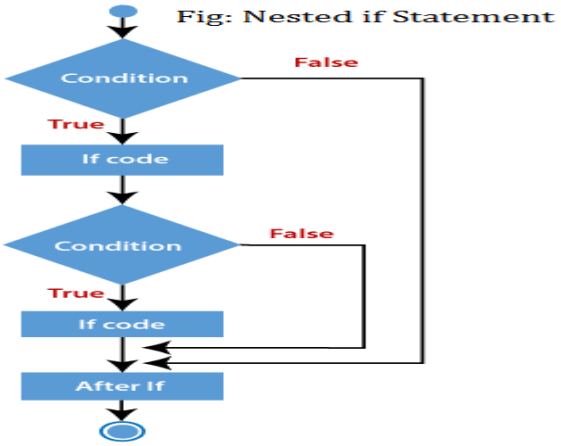


Fig: Nested if Statement

✓ **Example**

```
1. <?php
2.         $a = 34; $b = 56; $c = 45;
3.     if ($a < $b) {
4.         if ($a < $c) {
5.             echo "$a is smaller than $b and $c";
6.         }
7.     }
8. ?>
```

Output:

34 is smaller than 56 and 45

5. Switch:

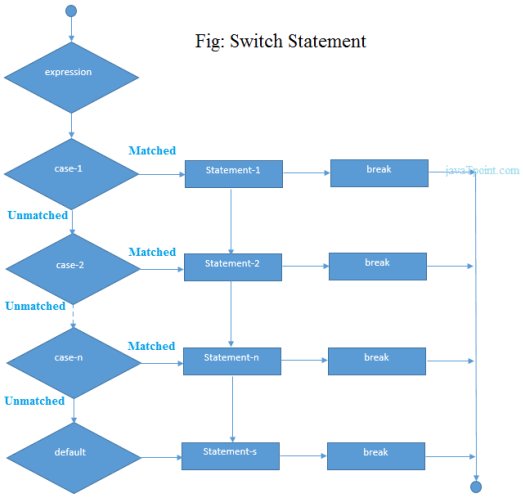


Fig: Switch Statement

- ✓ PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.
- ✓ Syntax:

```
1. switch(expression){
2. case value1:
3.  //code to be executed
4.  break;
5. case value2:
6.  //code to be executed
7.  break;
8. .....
9. default:
10. code to be executed if all cases are not matched; }
```

- ✓ **Example:**

```
1. <?php
2. $num=20;
3. switch($num){
4. case 10:
5. echo("number is equals to 10");
6. break;
7. case 20:
8. echo("number is equal to 20");
9. break;
10. case 30:
11. echo("number is equal to 30");
12. break;
13. default:
14. echo("number is not equal to 10, 20 or 30"); }
15. ?>
```

Output:

```
number is equal to 20
```

❖ Doing Repetitive task with looping:

➢ PHP for Loop:

- ✓ PHP for loop can be used to traverse set of code for the specified number of times. It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.It allows users to put all the loop related statements in one place.
- ✓ Syntax:

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

- ✓ Example:

```
<?php
for($n=1;$n<=10;$n++){
echo "$n<br/>";
}
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

➢ PHP While Loop:

- ✓ PHP while loop can be used to traverse set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of loop.
- ✓ It should be used if the number of iterations is not known. The while loop is also called an **Entry control loop** because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.

- ✓ Syntax

```
1. while(condition){
2. //code to be executed
3. }
```

Alternative Syntax

```
1. while(condition):
2. //code to be executed
3.
4. endwhile;
```

- ✓ Example

```
1. <?php
2. $n=1;
3. while($n<=10){
4. echo "$n<br/>";
5. $n++;
6. }
7. ?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

➢ PHP do-while loop:

- ✓ PHP do-while loop can be used to traverse set of code like php while loop. The PHP do-while loop is guaranteed to run at least once.
- ✓ The PHP do-while loop is used to execute a set of code of the program several times. If you have to execute the loop at least once and the number of iterations is not even fixed, it is recommended to use the **do-while** loop.
- ✓ It executes the code at least one time always because the condition is checked after executing the code.
- ✓ The do-while loop is very much similar to the while loop except the condition check. The main difference between both loops is that while loop checks the condition at the beginning, whereas do-while loop checks the condition at the end of the loop.
- ✓ Syntax

```
1. do{
2. //code to be executed
3. }while(condition);
```

- ✓ Example

```
1. <?php
2. $n=1;
3. do{
4. echo "$n<br/>";
5. $n++;
6. }while($n<=10);
7. ?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

➢ PHP foreach loop:

- ✓ The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.
- ✓ The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.
- ✓ In foreach loop, we don't need to increment the value.
- ✓ Syntax

```
1. foreach ($array as $value) {
2.     //code to be executed
3. }
```

There is one more syntax of foreach loop.

Syntax

```
1. foreach ($array as $key => $element) {
2.     //code to be executed
3. }
```

- ✓ **Example: PHP program to print array elements using foreach loop.**

```
1. <?php
2.     //declare array
3.     $season = array ("Summer", "Winter", "Autumn", "Rainy");
4.
5.     //access array elements using foreach loop
6.     foreach ($season as $element) {
7.         echo "$element";
8.         echo "</br>";
9.     }
10. ?>
```

Output:

```
Summer
Winter
Autumn
Rainy
```

❖ Mixing Decisions and looping with HTML:

- PHP will only processes things that are enclosed within one of its valid code blocks (such as <?php and ?>).

- **PHP Conditions for HTML Code:**
  - ✓ Let's consider we only want PHP to display a certain html code if condition is true then we can do that using the following syntax:

```
<?php
if(conditions)
{
?>
... HTML CODE ...
<?php
}
?>
```

  - ✓ Although this may be confusing, it is important to remember that how PHP will process this code.
  - ✓ A special syntax is provided for instances where PHP is being used simply to control the output of standard HTML code:

```
<?php if(conditions): ?>
... HTML CODE ...
<?php endif; ?>
```

  - ✓ Beyond simple if statements, most control structures provide an alternative syntax that allows us to embed PHP code within standard HTML quickly and easily.

```
<?php while(conditions) : ?>
... HTML CODE ...
<?php endwhile; ?>
```

  - ✓ And an identical syntax for an embedded for loop:

```
<?php for(init;conditions;increment) : ?>
... HTML CODE ...
<?php endfor; ?>
```

  - ✓ Example:

1	2	3	4	5	6
no	no	no	no	yes	yes

Let’s code

```
<html>
<body>
<table>
<tr>
<td align="center">1</td>
<td align="center">2</td>
<td align="center">3</td>
<td align="center">4</td>
<td align="center">5</td>
<td align="center">6</td>
</tr>
<tr>
<td align="center">no</td>
<td align="center">no</td>
<td align="center">no</td>
<td align="center">no</td>
<td align="center">yes</td>
<td align="center">yes</td>
</tr>
</table>
</body>
</html>
```

- ✓ PHP code:

```
<html>
<body>
<table>
<tr>
<?php for($l = 1; $l <=6; $l++) : ?>
<td align="center"><?=$l?></td>
<?php endfor; ?>
</tr>
<tr>
<?php for($l = 1; $l <=6; $l++) : ?>
<td align="center">
<?php if($l >= 5)  {
echo "yes";
} else {
echo "no";
}
?>
</td>
<?php endfor; ?>
</tr>
</table>
</body>
</html>
```

INSTITUTE OF MANAGEMENT AND INFORMATION TECHNOLOGY

Subject- PHP & MYSQL(MCA 406E)

RACHANA BEHERA

Email-id: rachanabehera2792@gmail.com

UNIT 2

Function:

❖ What is a function?

- A function is a set of statements that performs a particular function and optionally returns a value.
- PHP function is a piece of code that can be reused many times. It can take input as argument list and return value.
- Advantage of PHP Functions:

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

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

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

❖ Define a function:

- There are thousands of built-in functions in PHP. In PHP, we can define Conditional function, Function within Function and Recursive function also.
- 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.
  - ✓ Create a User Defined Function in PHP
  - ✓ A user-defined function declaration starts with the word function:
  - ✓ **Syntax:**

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

  - ✓ **Example:**

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

Output:

Hello world!

➢ PHP Function Arguments:

- ✓ We can pass the information in PHP function through arguments which is separated by comma.
- ✓ PHP supports Call by Value (default), Call by Reference, Default argument values and Variable-length argument list.
- ✓ **Example to pass single argument in PHP function:**

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

- ✓ **Example to pass two argument in PHP function:**

```
<?php
function sayHello($name,$age)
{
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

❖ Call by value:

- ➢ PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.
- ➢ Example 1:

```
<?php
function adder($str2)
{
   $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

```
Hello
```

✓ Example 2:

```
<?php
function increment($i)
{
   $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output:

10

❖ Call by reference:

- In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.
- Example 1:

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
echo $str;
?>
```

Output:

This is Call By Reference

- Example 2:

```
<?php
function increment(&$i)
{
   $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output:

11

❖ Recursive function:

- PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

- Example 1: Printing number

```
<?php
function display($number)
{
   if($number<=5){
    echo "$number <br/>";
    display($number+1);
   }
}
display(1);
?>
```

Output:

1
2
3
4
5

- Example 2 : Factorial Number

```
<?php
function factorial($n)
{
   if ($n < 0)
      return -1; /*Wrong value*/
   if ($n == 0)
      return 1; /*Terminating condition*/
   return ($n * factorial ($n -1));
}
echo factorial(5);
?>
```

Output:

120

❖ String Creating and accessing:

- PHP string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set and so that it does not offer native Unicode support.
- There are 4 ways to specify a string literal in PHP.

1. Single Quoted:

- We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP. For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\). All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.
- Example:

```
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

Output:

Hello text within single quote

2. Double Quoted:

- In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.
- Example:

```
<?php
$str="Hello text within double quote";
echo $str;
?>
```

Output:

Hello text within double quote

3. Heredoc:

- Heredoc syntax (<<<) is the third way to delimit strings. In Heredoc syntax, an identifier is provided after this heredoc <<< operator, and immediately a new line is started to write any text. To close the quotation, the string follows itself and then again that same identifier is provided. That closing identifier must begin from the new line without any whitespace or tab.It must contain only alphanumeric characters and underscores, and must start with an underscore or a non-digit character.
- Example:

```
<?php
   $str = <<<Demo
It is a valid example
Demo;      //Valid code as whitespace or tab is not valid before closing identifier
echo $str;
?>
```

Output:

It is a valid example

4. Newdoc:

- Newdoc is similar to the heredoc, but in newdoc parsing is not done. It is also identified with three less than symbols <<< followed by an identifier. But here identifier is enclosed in single-quote, e.g. <<<'EXP'. Newdoc follows the same rule as heredocs.
- The difference between newdoc and heredoc is that - Newdoc is a single-quoted stringwhereas heredoc is a double-quoted string.
- Example-1:

```
<?php
   $str = <<<'DEMO'
   Welcome to javaTpoint.
DEMO;
echo $str;
echo '</br>';
 echo <<< 'Demo'     // Here we are not storing string content in variable str.
   Welcome to javaTpoint.
Demo;
?>
```

Output:

Welcome to javaTpoint.
Welcome to javaTpoint.

❖ String Searching:

- The strchr() function searches for the first occurrence of a string inside another string.
- This function is an alias of the strstr() function.
- This function is binary-safe.
- This function is case-sensitive. For a case-insensitive search, use stristr() function.
- Syntax:

  strchr(string,search,before_search);

- Parameter Values:

Parameter	**Description**
string	Required. Specifies the string to search
search	Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number
before_search	Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the search parameter.

- Example:

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

Output:

Hello

❖ Replacing String:

- The str_replace() function replaces some characters with some other characters in a string.
- This function works by the following rules:
  - If the string to be searched is an array, it returns an array
  - If the string to be searched is an array, find and replace is performed with every array element
  - If both find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace
  - If find is an array and replace is a string, the replace string will be used for every find value
- This function is case-sensitive. Use the str_ireplace() function to perform a case-insensitive search. This function is binary-safe.
- Syntax:

  str_replace(find,replace,string,count)

- Parameter Values:

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

- Example:

```
<html>
<body>
<p>Search an array for the value "RED", and then replace it with "pink".</p>
<?php
$arr = array("blue","red","green","yellow");
print_r(str_replace("red","pink",$arr,$i));
echo "<br>" . "Replacements: $i";
?>
</body>
</html>
```

Output:

Search an array for the value "RED", and then replace it with "pink".

Array ([0] => blue [1] => pink [2] => green [3] => yellow)

Replacements: 1

❖ Formatting String:

- PHP features the versatile printf() and sprintf() functions that you can use to format string in many different ways.
- printf():
  The printf() function outputs a formatted string.
  Syntax:
  printf(format,arg1,arg2,arg++)

  Example:

```
<?php
$number=123;
printf("%f",$number);
?>
```

- sprintf():
  The sprintf() function writes a formatted string to a variable.
  Synatx:
  Sprintf(format,arg1,arg2,arg++)
  Example:

```
<?php
$number=123;
$txt=sprintf("%f",$number);
echo $txt;
?>
```

❖ String Related Library function:

1) strtolower():

- The strtolower() function returns string in lowercase letter.
- Syntax: **string strtolower (string $string)**
- Example:

```
<?php
$str="My name is Rachana";
$str=strtolower($str);
echo $str;
?>
```

Output:

my name is rachana

2) strtoupper():

- The strtoupper() function returns string in uppercase letter.
- Syntax: **string strtoupper (string $string)**
- Example:

```
<?php
$str="My name is Rachana";
$str=strtoupper($str);
echo $str;
?>
```

Output:

MY NAME IS RACHANA

3) ucfirst():

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

string ucfirst (string $str)

- Example:

```
<?php
$str="my name is Rachana";
$str=ucfirst($str);
echo $str;
?>
```

Output:

My name is Rachana.

4) lcfirst():

- The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.
- Syntax: **string lcfirst (string $str)**
- Example:

```
<?php
$str="MY name IS RACHANA";
$str=lcfirst($str);
echo $str;
?>
```

Output:

mY name IS RACHANA

5) ucwords():

- The ucwords() function returns string converting first character of each word into uppercase.
- Syntax:

  string ucwords (string $str)
- Example:

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

Output:

My Name Is Sonoo Jaiswal

6) strrev():

- The strrev() function returns reversed string.
- Syntax:

  string strrev (string $string)
- Example:

```
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

lawsiaj oonoS si eman ym

7) strlen():

- The strlen() function returns length of the string.
- Syntax:

  int strlen (string $string)
- Example:

```
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

24

ARRAY:

❖ Anatomy of an Array:

- PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.
- **Advantage of PHP Array:**

  Less Code: We don't need to define multiple variables.

  Easy to traverse: By the help of single loop, we can traverse all the elements of an array.

  Sorting: We can sort the elements of array.
- There are 3 types of array in PHP:
  1. Indexed Array
  2. Associative Array
  3. Multidimensional Array

❖ PHP Indexed Array:

- PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.
- PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as numeric array.
- There are two ways to define indexed array:

1st way:

```
$size=array("Big","Medium","Short");
```

2nd way:

```
$size[0]="Big";
$size[1]="Medium";
$size[2]="Short";
```

Example1:

```
<?php
$size=array("Big","Medium","Short");
echo "Size: $size[0], $size[1] and $size[2]";
?>
```

Output:

Size: Big, Medium and Short

Example 2:

```
<?php
$size[0]="Big";
$size[1]="Medium";
$size[2]="Short";
echo "Size: $size[0], $size[1] and $size[2]";
?>
```

Output:

Size: Big, Medium and Short

➢ Traversing PHP Indexed Array:

- We can easily traverse array in PHP using foreach loop.
- Example:

```
<?php
$size=array("Big","Medium","Short");
foreach( $size as $s )
{
  echo "Size is: $s<br />";
}
?>
```

Output:

Size is: Big
Size is: Medium
Size is: Short

➢ Count Length of PHP Indexed Array:

- PHP provides count() function which returns length of an array.
- Example:

```
<?php
$size=array("Big","Medium","Short");
echo count($size);
?>
```

Output:

3

❖ PHP Associative Array:

➢ PHP allows you to associate name/label with each array elements in PHP using => symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number.

➢ There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
```

Example 1:

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Output:

Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000

Example 2:

```
<?php
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Output:

Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000

➢ Traversing PHP Associative Array:

By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

Example:

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
foreach($salary as $k => $v) {
echo "Key: ".$k." Value: ".$v."<br/>";
}
?>
```

Output:

Key: Sonoo Value: 550000
Key: Vimal Value: 250000
Key: Ratan Value: 200000

❖ PHP Multidimensional Array:

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

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

- PHP code:

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

for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo $emp[$row][$col]."  ";
  }
  echo "<br/>";
}
?>
```

Output:

1 sonoo 400000
2 john 500000
3 rahul 300000

❖ Some useful Library function:

is_array():

- The is_array() function checks whether a variable is an array or not.
- This function returns true (1) if the variable is an array, otherwise it returns false/nothing.
- Syntax: **is_array(variable);**
- Example:

```
<html>
<body>
<?php
$a = "Hello";
echo "a is " . is_array($a) . "<br>";
$b = array("red", "green", "blue");
echo "b is " . is_array($b) . "<br>";
$c = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "c is " . is_array($c) . "<br>";
$d = "red, green, blue";
echo "d is " . is_array($d) . "<br>";
?>
</body>
</html>
```

Output: **a is**
b is 1
c is 1
d

count():

- This function is used to return the number of elements in an array.
- Syntax: **count(variable)**
- Example:

```
<html>
<body>
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
</body>
</html>
```

Output**: 3**

sort():

- Sort the elements of the array in ascending order.
- Synatx: **sort(variable)**
- Example:

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

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

Output:

BMW
Toyota
Volvo

shuffle():

- The shuffle() function randomizes the order of the elements in the array.
- This function assigns new keys for the elements in the array. Existing keys will be removed (See Example below).
- Syntax: **shuffle(array)**
- Example:

```
<html>
<body>
<?php
$my_array = array("red","green","blue","yellow","purple");
shuffle($my_array);
print_r($my_array);
?>
<p>Refresh the page to see how shuffle() randomizes the order of the elements in the array.</p>
</body>
</html>
```

Output:

Array ([0] => green [1] => purple [2] => yellow [3] => red [4] => blue)
Refresh the page to see how shuffle() randomizes the order of the elements in the array.

compact():

- The compact() function creates an array from variables and their values.
- Any strings that does not match variable names will be skipped.
- Syntax: **compact(var1, var2...)**
- Example:

```
<html>
<body>
<?php
$firstname = "Peter";
$lastname = "Griffin";
$age = "41";
$result = compact("firstname", "lastname", "age");
print_r($result);
?>
</body>
</html>
```

Output:

Array ([firstname] => Peter [lastname] => Griffin [age] => 41)

reset():

- The reset() function moves the internal pointer to the first element of the array.
- Related methods:

  current() - returns the value of the current element in an array

  end() - moves the internal pointer to, and outputs, the last element in the array

  next() - moves the internal pointer to, and outputs, the next element in the array

  prev() - moves the internal pointer to, and outputs, the previous element in the array

  each() - returns the current element key and value, and moves the internal pointer forward
- Syntax: **reset(array)**
- Example:

```
<html>
<body>
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
echo current($people) . "<br>"; // The current element is Peter
echo next($people) . "<br>"; // The next element of Peter is Joe
echo current($people) . "<br>"; // Now the current element is Joe
echo prev($people) . "<br>"; // The previous element of Joe is Peter
echo end($people) . "<br>"; // The last element is Cleveland
echo prev($people) . "<br>"; // The previous element of Cleveland is Glenn
echo current($people) . "<br>"; // Now the current element is Glenn
echo reset($people)."<br>";//Moves the internal pointer to the first element of the array(peter)
echo next($people) . "<br>" . "<br>"; // The next element of Peter is Joe
print_r (each($people));
?>
```

Output:

```
Peter
Joe
Joe
Peter
Cleveland
Glenn
Glenn
Peter
Joe
Array ( [1] => Joe [value] => Joe [0] => 1 [key] => 1 )
```

INSTITUTE OF MANAGEMENT AND INFORMATION TECHNOLOGY

Subject- PHP & MYSQL(MCA 406E)

RACHANA BEHERA

Email-id: rachanabehera2792@gmail.com

UNIT 3

Handling Html Form with Php:

- The main way that website users interact with PHP and MySQL is through the use of HTML forms.
- Handling forms is a multipart process.
- First a form is created, into which a user can enter the required details. This data is then sent to the web server, where it is interpreted, often with some error checking. If the PHP code identifies one or more fields that require re-entering, the form may be redisplayed with an error message.
- When the code is satisfied with the accuracy of the input, it takes some action that usually involves the database, such as entering details about a purchase.
- To build a form, you must have at least the following elements:
  - An opening <form> and closing </form> tag
  - A submission type specifying either a Get or Post method
  - One or more input fields
  - The destination URL to which the form data is to be submitted

❖ Capturing Form Data:

- A common and simple way of gathering data is through HTML Forms. Forms are containers for user input and can contain any number of different input types. The HTML form element requires a few parameters to work properly.
- **Action:** this should point to the page that is meant to process the collected data. As soon as the form is submitted, the browser is redirected to this location, along with all your data.
- **Method:** this is the method of transportation. There are two choices here: 'GET' and 'POST'.
- Both GET and POST are treated as $_GET and $_POST. These 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.
- $_GET is an array of variables passed to the current script via the URL parameters.
- $_POST is an array of variables passed to the current script via the HTTP POST method.

➢ $_GET:

- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL).
- GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.
- GET should NEVER be used for sending passwords or other sensitive information!

- Example:

```
<html>
<body>
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

"welcome_get.php" :

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```

➢ $_POST:

- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.
- Example:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

“welcome.php”

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

❖ Input Types:

- HTML forms are very versatile and allow you to submit a wide range of input types, from text boxes and text areas to checkboxes, radio buttons, and more.
- Text boxes: The input type you will probably use most often is the text box. It accepts a wide range of alphanumeric text and other characters in a single-line box. The general format of a text box input is as follows:
  <input type="text" name="*name*" size="*size*" maxlength="*length*" value="*value*">
- Text areas: When you need to accept input of more than a short line of text, use a text area. This is similar to a text box, but, because it allows multiple lines, it has some different attributes. Its general format looks like this:
  <textarea name="*name*" cols="*width*" rows="*height*" wrap="*type*"></textarea>
- Checkboxes: When you want to offer a number of different options to a user, from which he can select one or more items, checkboxes are the way to go. Here is the format to use:
  <input type="checkbox" name="*name*" value="*value*" checked="checked">
  If you include the checked attribute, the box is already checked when the browser is displayed.
- Radio button: Radio buttons are named after the push-in preset buttons found on many older radios, where any previously depressed button pops back up when another is pressed. They are used when you want only a single value to be returned from a selection of two or more options. All the buttons in a group must use the same name and, because only a single value is returned, you do not have to pass an array.
  <input type=”radio” name=”name” value=”value”>
- Hidden fields: Sometimes it is convenient to have hidden form fields so that you can keep track of the state of form entry. For example, you might wish to know whether a form has already been submitted. You can achieve this by adding some HTML in your PHP code, such as the following:
  echo '<input type="hidden" name="submitted" value="yes">'
  This is a simple PHP echo statement that adds an input field to the HTML form.
- <select>: The <select> tag lets you create a drop-down list of options, offering either single or multiple selections. It conforms to the following syntax:
  <select name="*name*" size="*size*" multiple="multiple">
  The attribute size is the number of lines to display. Clicking on the display causes a list to drop down, showing all the options. If you use the multiple attribute, a user can select multiple options from the list by pressing the Ctrl key when clicking.
- Example:

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
File: login.php
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];
echo "Welcome: $name, your password is: $password";
?>
```

❖ Dealing with Multi-value filed

- Form fields can send multiple values, rather than a single value.
- The trick is to add square brackets ([]) after the field name in your HTML form. When PHP engine sees a submitted form field name with square brackets at the end, it creates a nested array of values within the $_GET or $_POST and $_REQUEST superglobal array, rather than a single value.
- You can then pull the individual values out of that nested array. So you might create a multi-select list control as follows:
  <select name="mySelection[]" id="mySelection" size="3" multiple="multiple"> ... </select>
- Example:

```
<html>
<body>
<form action="demo.php"method="post">
<fieldset style="width:200px">
<legend>Personal Info</legend>
First Name<br>
<input type="text"name="txtfname"><br>
Last name<br>
<input type="text"name="txtlname"><br>
</fieldset>
<fieldset style="width:200px">
<legend>Platform Interested</legend>
<input type="checkbox"name="chkplatform[]"value="PHP">PHP<br>
<input type="checkbox"name="chkplatform[]"value="JAVA">JAVA<br>
<input type="checkbox"name="chkplatform[]"value="C">C<br>
</fieldset>
<input type="submit" name="btnsave"value="save">
<input type="reset" name="btnreset"value="Reset">
</form>
</body>
</html>
```

demo.php:

```
<?php
if(isset($_POST["btnsave"]))
{
        $firstname=$_POST["txtfname"];
        $lastname=$_POST["txtlname"];
        $platform=$_POST["chkplatform"];
        echo"<h1>First Name=$firstname</h1>";
        echo"<h1>Last Name=$lastname</h1>";
        foreach($platform as $platforms=>$p)
        { echo"<b>platform interested:$p</b><br>";}
}
else
{
        echo"Form data is not submitted";
}
?>
```

❖ Generating File uploaded form:

- A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.
- Information in the phpinfo.php page describes the temporary directory that is used for file uploads as upload_tmp_dir and the maximum permitted size of files that can be uploaded is stated as upload_max_filesize. These parameters are set into PHP configuration file php.ini.
- The process of uploading a file follows these steps –
  - The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
  - The user clicks the browse button and selects a file to upload from the local PC.
  - The full path to the selected file appears in the text filed then the user clicks the submit button.
  - The selected file is sent to the temporary directory on the server.
  - The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
  - The PHP script confirms the success to the user.
  - As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail.
  - An uploaded file could be a text file or image file or any document.

Working with file and Directories:

❖ Understanding file & directory:

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.

❖ Opening and closing a file:

PHP fopen():

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

  resource fopen (string $filename , string $mode [, bool $use_include_path = false [, resource $context]])
- PHP Open File Mode:

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

x	Creates and opens file in write-only mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in read-write mode.
c	Opens file in write-only mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file
c+	It is same as c but it opens file in read-write mode.

- Example:

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

PHP Close File :

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

```
fclose ( open function variable name)
```

- Example:

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

PHP Read File:

- 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()

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

  Syntax:

  string fread (resource filename , filesize or int $length)
  where $length represents length of byte to be read.

  Example:

```
<?php
$filename = "c:\\file1.txt";
$fp = fopen($filename, "r");//open file in read mode
$contents = fread($fp, filesize($filename));//read file
echo "<pre>$contents</pre>";//printing data of file
fclose($fp);//close file
?>
```

  Output:

```
this is first line
this is another line
this is third line
```

➢ fgets():

The PHP fgets() function is used to read single line from the file.

Syntax:

string fgets (resource filename [, int $length])

Example:

```
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
echo fgets($fp);
fclose($fp);
?>
```

Output

this is first line

➢ 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 filename)

Example:

```
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
while(!feof($fp)) {
  echo fgetc($fp);
}
fclose($fp);
?>
```

Output

this is first line this is another line this is third line

➢ feof():

This function checks if the "end-of-file" (EOF) has been reached for an open file. This function is useful for looping through data of unknown length.

Syntax:

feof(file)

Example:

```
<?php
$file = fopen("test.txt", "r");
//Output lines until EOF is reached
while(! feof($file)) {
  $line = fgets($file);
  echo $line. "<br>";
}
fclose($file);
?>
```

PHP Write File

- PHP fwrite() and fputs() functions are used to write data into file. To write data into file, you need to use w, r+, w+, x, x+, c or c+ mode.
- The PHP fwrite() function is used to write content of the string into file.
- Syntax:

  int fwrite (resource filename , string $string [, int $length])
- Example:

```
<?php
$fp = fopen('data.txt', 'w');//opens file in write-only mode
fwrite($fp, 'welcome ');
fwrite($fp, 'to php file write');
fclose($fp);
echo "File written successfully";
?>
```

  Output: data.txt

  welcome to php file write
- **PHP Append to File**

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

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

  Example:

```
<?php
$fp = fopen('data.txt', 'a');//opens file in append mode
fwrite($fp, ' this is additional text ');
fwrite($fp, 'appending data');
fclose($fp);
echo "File appended successfully";
?>
```

  Output: data.txt

  welcome to php file write this is additional text appending data

❖ Coping, renaming and deleting a file:

copy( ) Function:

- The copy() function in PHP is an inbuilt function which is used to make a copy of a specified file. It makes a copy of the source file to the destination file and if the destination file already exists, it gets overwritten. The copy() function returns true on success and false on failure.
- Syntax:

  bool copy ($source, $dest)
- Example 1:

```
<?php
 echo copy("textfile1.txt", "textfile2.txt");
?>
```

  Output:

  1

- Example 2:

```
<?php
$srcfile = 'textfile1.txt';
$destfile = 'textfile2.txt';
if (!copy($srcfile, $destfile))
{ echo "File cannot be copied. \n"; }
else
{  echo "File has been copied."; }
?>
```

Output:

File has been copied.

rename( ) Function:

- The rename() function in PHP is an inbuilt function which is used to rename a file or directory.
- It makes an attempt to change an old name of a file or directory with a new name specified by the user and it may move between directories if necessary.
- If the new name specified by the user already exists, the rename() function overwrites it. The old name of the file and the new name specified by the user are sent as parameters to the rename() function and it returns True on success and a False on failure.
- Syntax:

```
rename(oldname, newname, context)
```

- Example 1:

```
<?php
$old_name = "ofile.txt" ;
$new_name = "nfile.txt" ;
rename( $old_name, $new_name) ;
?>
```

Output:

1

- Example 2:

```
<?php
$old_name = "ofile.txt" ;
$new_name = "nfile.txt" ;
if (!rename($old_name, $new_name))
{
 echo "File cannot be renamed. \n";
}
else
{
echo "File has been renamed.";
}
?>
```

Output:

File has been renamed.

Delete a file:

- To delete a file by using PHP is very easy. Deleting a file means completely erase a file from a directory so that the file is no longer exist. PHP has an unlink() function that allows to delete a file. The PHP unlink() function takes two parameters $filename and $context.
- Syntax:

  unlink($filename, $context);
- Example:

```
<?php
$file="text2.txt";
if(!unlink($file))
{
        echo"File cannot be deleted due to error";
}
else
{
        echo"File has been deleted";
}
?>
```

Output:

File has been deleted.

❖ Working with directories:

getcwd( ):

- The full form of getcwd is "Get Current Working Directory", the function getcwd() is used to get the name of the current working directory, it does not accept any parameter and returns the current working directory.
- Syntax:

  getcwd();
- It does not accept any parameter.
- Example: PHP code to get the name of the current working directory

```
<?php
$result = getcwd();
echo "current working directory is: ".$result."<br/>";
?>
```

Output:

current working directory is: C:\xampplite\htdocs\tutorial

mkdir() :

- The full form of mkdir is "Make Directory", the function mkdir() is used to create a directory.
- Syntax:

  mkdir(dir_path, access_mode, recursive, context);

- Parameter(s):

  dir_path – It defines the path to the directory, where we want to create a directory.

  access_mode – It is an optional parameter; its default value is 0777 that stands for the widest possible access. There are 4 values to be set for the access mode,

  - The first value should be 0
  - The second value sets the permission for the owner
  - The third value sets the permission for the owner's user group
  - The fourth value sets the permission for everybody else
  - The values are 1 for execute permission, 2 for write permission, 4 for reading permission, we can add values to set the specific permissions, for example, 1+2+4 = 7 = permission for executing, write and read.
  - The access_mode parameter is ignored on Windows system.

  recursive – It is also an optional parameter; It defines the recursive mode.

  context - It is also an optional parameter; It sets the context (a set of options that can modify the behavior of a stream) of the file handling.

- Example: PHP code to create directory

```
<?php
$result = mkdir("c:/xampplite/htdocs/tutorial/folder1");
if($result==true)
        echo"directory created successfully";
else
        echo "directory is not created";
?>
```

Output:

directory created successfully

chdir():

- The full form of chdir is "Change Directory", the function chdir() is used to change the current working directory.
- Syntax:

```
mkdir(directory);
```

directory – It defines the new directory.

- Example: PHP code to change the directory

```
<?php
$result=getcwd();
echo "current working directory is: ".$result."<br/>";
chdir("c:/xampplite/htdocs/tutorial/folder1");
$result=getcwd();
echo "current working directory is: ".$result."<br/>";
?>
```

Output:

current working directory is: C:\xampplite\htdocs\tutorial
current working directory is: C:\xampplite\htdocs\tutorial\folder1

is_dir():

- The full form of is_dir is "Is Directory", the function is_dir() is used to check whether a file system is a directory or whether a directory exists or not.
- Syntax:

  is_dir(directory);
- Example: PHP code to check whether a directory exists or not

```
<?php
mkdir("c:/xampplite/htdocs/tutorial/folder2");
if(is_dir("c:/xampplite/htdocs/tutorial/folder2"))
        echo"this folder is exists";
else
        echo "folder is not exists";
?>
```

Output:

this folder is exists

scandir():

- The full form of scandir is "Scan Directory", the function scandir() is used to get the list of the files and directories available in the specified directory.
- Syntax:

  scandir(directory, sorting_order, context);
- Parameter(s):

  directory – It specifies the directory name (or path) from there we have to get the list of the files and directories

  sorting – It is an optional parameter; its default value is o (alphabetically sorting order). 1 can be used to descending order.

  context – It is an optional parameter; it is used to specify the context (a set of options that can modify the behavior of the stream) to the directory handle.
- Example: PHP code to get and print the list of files and directories of a given directory

```
<?php
$path="e:/";
$arr1=scandir($path);
print_r($arr1);
?>
```

Output:

```
Array
(
   [0] => .
   [1] => ..
   [2] => folder1
   [3] => folder2
   [4] => folder3
   [5] => main.php
)
```

rmdir():

- It is used to remove directory or folder.
- Syntax:

```
rmdir(directory);
```

- Example:

```
<?php
$path="c:/xampplite/htdocs/tutorial/folder1";
rmdir($path);
?>
```

❖ File Uploading & Downloading:

- PHP allows you to upload single and multiple files through few lines of code only.
- PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.
- **PHP $_FILES:** The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file. Here, we are assuming that file name is filename.

  $_FILES['filename']['name']: returns file name.

  $_FILES['filename']['type']: returns MIME type of the file.

  $_FILES['filename']['size']: returns size of the file (in bytes).

  $_FILES['filename']['tmp_name']: returns temporary file name of the file which was stored on the server.

  $_FILES['filename']['error']: returns error code associated with this file.
- **move_uploaded_file():** Thisfunction moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.
- **Example:**

```
<?php
echo <<<_END
<html><head><title>PHP Form Upload</title></head><body>
<form method='post' action='upload.php' enctype='multipart/form-data'>
Select File: <input type='file' name='filename' size='10'>
<input type='submit' value='Upload'>
</form>
_END;
if ($_FILES)
{
$name = $_FILES['filename']['name'];
move_uploaded_file($_FILES['filename']['tmp_name'], $name);
echo "Uploaded image '$name'<br><img src='$name'>";
}
echo "</body></html>";
?>
```

File Upload:

```
<html>
<head>
<title>PHP File Uploading Form</title>
</head>
<body>
<form action="uploader.php" method="post" enctype="multipart/form-data">
<input type="file"name="uploadFile"/>
<input type="submit"name="submitBtn"value="click to uplaod file"/>
</form>
</body>
</html>
```

uploader.php:

```
<?php
if(isset($_POST['submitBtn']))
{
        //print_r($_FILES);
        $file_name=$_FILES['uploadFile']['name'];
        $file_size=$_FILES['uploadFile']['size'];
        $file_temp=$_FILES['uploadFile']['tmp_name'];
        $file_type=$_FILES['uploadFile']['type'];
        $file_ext=strtolower(end(explode('.',$file_name)));
        //$allowed_ext=array('jpeg','jpg','png','gif','txt');
        if($file_ext)
        {
                move_uploaded_file($file_temp,'uploaded_files/'.$file_name);
                echo 'File Uploaded successfully';
        }
        else
        {
                echo $file_ext.  ' File type is not allowed here.';
        }
}
?>
```

PHP Files Downloading:

```
<?php
$filePath='uploaded_files/006.jpg';
if(file_exists($filePath))
{
        header('Content-Description:File Transfer');
        header('Content-Type:application/octet-stream');
        header('Content-Disposition:attachment;filename="'.basename($filepPath).'"');
        header('expires:0');
        header('Cache-Control:must_revalidated');
        header('Progma:Public');
        header('content-Length:'.filesize($filePath));
        flush();
        readfile($filePath);
        exit;
}
?>
```

INSTITUTE OF MANAGEMENT AND INFORMATION TECHNOLOGY

Subject- PHP & MYSQL(MCA 406E)

RACHANA BEHERA

Email-id: rachanabehera2792@gmail.com

UNIT 4

Session and Cookie:

❖ Introduction to Session Control:

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL.
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

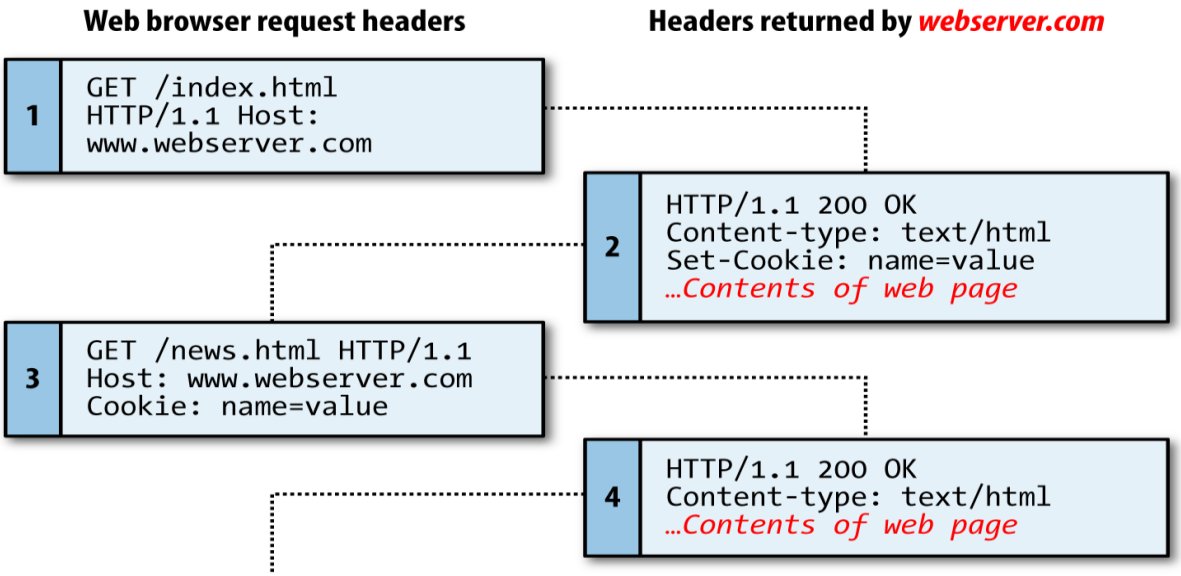
❖ Session Functionality:

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL.
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

❖ What is a Cookie:

- A *cookie* is an item of data that a web server saves to your computer's hard disk via a web browser.
- It can contain almost any alphanumeric information (as long as it's under 4 KB) and can be retrieved from your computer and returned to the server.
- Common uses include session tracking, maintaining data across multiple visits, holding shopping cart contents, storing login details, and more.
- Because of their privacy implications, cookies can be read only from the issuing domain.
- In other words, if a cookie is issued by, for example, *oreilly.com*, it can be retrieved only by a web server using that domain. This prevents other websites from gaining access to details for which they are not authorized.
- Because of the way the Internet works, multiple elements on a web page can be embedded from multiple domains, each of which can issue its own cookies. When this happens, they are referred to as *third-party cookies*. Most commonly, these are created by advertising companies in order to track users across multiple websites.

- Because of this, most browsers allow users to turn cookies off either for the current server's domain, third-party servers, or both. Fortunately, most people who disable cookies do so only for third-party websites.
- Cookies are exchanged during the transfer of headers, before the actual HTML of a web page is sent, and it is impossible to send a cookie once any HTML has been transferred. Therefore, careful planning of cookie usage is important.
- A browser/server request/response dialog with cookies:



- This exchange shows a browser receiving two pages:
  1. The browser issues a request to retrieve the main page, *index.html*, at the website *http://www.webserver.com*. The first header specifies the file, and the second header specifies the server.
  2. When the web server at *webserver.com* receives this pair of headers, it returns some of its own. The second header defines the type of content to be sent (text/html), and the third one sends a cookie of the name *name* and with the value *value*. Only then are the contents of the web page transferred.
  3. Once the browser has received the cookie, it will then return it with every future request made to the issuing server until the cookie expires or is deleted. So, when the browser requests the new page */news.html*, it also returns the cookie *name* with the value *value*.
  4. Because the cookie has already been set, when the server receives the request to send */news.html*, it does not have to resend the cookie, but just returns the requested page.

❖ Setting Cookies with PHP:

- Setting a cookie in PHP is a simple matter. As long as no HTML has yet been transferred, you can call the setcookie function.
- Syntax:

  setcookie (name, value, expire, path, domain, secure, httponly);
- *The setcookie parameters:*

  name: The name of the cookie. This is the name that your server will use to access the cookie on subsequent browser requests (Ex: username)

  value: The value of the cookie, or the cookie's contents. This can contain up to 4 KB of alphanumeric text (Ex: Hannah)

  expire (*Optional.*): Unix timestamp of the expiration date. Generally, you will probably use time() plus a number of seconds. If not set, the cookie expires when the browser closes (Ex: time() + 2592000)

path (*Optional.*): The path of the cookie on the server. If this is a / (forward slash), the cookie is available over the entire domain, such as *www.webserver.com*. If it is a subdirectory, the cookie is available only within that subdirectory. The default is the current directory that the cookie is being set in, and this is the setting you will normally use (Ex: /)

domain (*Optional.*): The Internet domain of the cookie. If this is *.webserver.com*, the cookie is available to all of *webserver.com* and its subdomains, such as *www.webserver.com* and *images.webserver.com*. If it is *images.webserver.com*, the cookie is available only to *images.webserver.com* and its subdomains such as *sub.images.webserver.com*, but not, say, to *www.webserver.com*. (Ex: .webserver.com)

secure (*Optional.*): Whether the cookie must use a secure connection (*https://*). If this value is TRUE, the cookie can be transferred only across a secure connection. The default is FALSE. (Ex: FALSE)

httponly (*Optional*; implemented since PHP version 5.2.0.): Whether the cookie must use the HTTP protocol. If this value is TRUE, scripting languages such as JavaScript cannot access the cookie. (Not supported in all browsers.) The default is FALSE. (Ex: FALSE)

- So, to create a cookie with the name username and the value Hannah that is accessible across the entire web server on the current domain, and will be removed from the browser's cache in seven days, use the following:

setcookie('username', 'Hannah', time() + 60 * 60 * 24 * 7, '/');

❖ Accessing a Cookies:

- Reading the value of a cookie is as simple as accessing the $_COOKIE system array. For example, if you wish to see whether the current browser has the cookie called username already stored and, if so, to read its value, use the following:

if (isset($_COOKIE['username'])) $username = $_COOKIE['username'];

- Note that you can read a cookie back only after it has been sent to a web browser.
- This means that when you issue a cookie, you cannot read it in again until the browser reloads the page (or another with access to the cookie) from your website and passes the cookie back to the server in the process.
- Example:

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

Output: **Cookie 'user' is set**

Value is: Alex Porter

❖ Deleting Cookies:

- To delete a cookie, you must issue it again and set a date in the past. It is important for all parameters in your new setcookie call except the timestamp to be identical to the parameters when the cookie was first issued; otherwise, the deletion will fail. Therefore, to delete the cookie created earlier, you would use the following:

  setcookie('username', 'Hannah', time() - 2592000, '/');

- As long as the time given is in the past, the cookie should be deleted. However, I have used a time of 2,592,000 seconds (one month) in the past in case the client computer's date and time are not correctly set.
- Example:

```
<?php
// set the expiration date to one hour ago
setcookie ("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

Output:

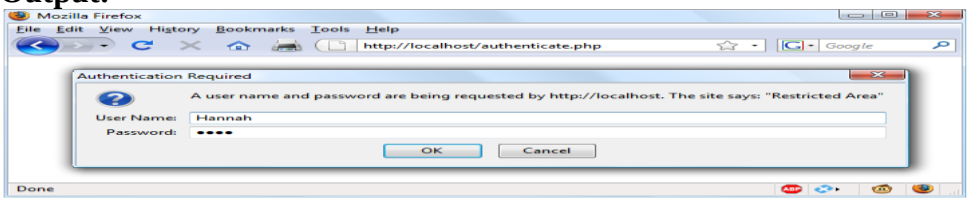
Cookie 'user' is deleted.

❖ HTTP Authentication:

- HTTP authentication uses the web server to manage users and passwords for the application. It's adequate for most applications that ask users to log in, although some applications have specialized needs or more stringent security requirements that call for other techniques.
- To use HTTP authentication, PHP sends a header request asking to start an authentication dialog with the browser. The server must have this feature turned on in order for it to work, but because it's so common, your server is likely to offer the feature.
- After entering your URL into the browser or visiting via a link, the user will see an Authentication Required" prompt pop up, requesting two fields: User Name and Password
- Example 1: PHP authentication

```
<?php
if (isset($_SERVER['PHP_AUTH_USER']) &&
    isset($_SERVER['PHP_AUTH_PW']))
{
echo "Welcome User: " . $_SERVER['PHP_AUTH_USER'] .
    " Password: " . $_SERVER['PHP_AUTH_PW'];
}
else
{
header('WWW-Authenticate: Basic realm="Restricted Section"');
header('HTTP/1.0 401 Unauthorized');
die("Please enter your username and password");
}
?>
```

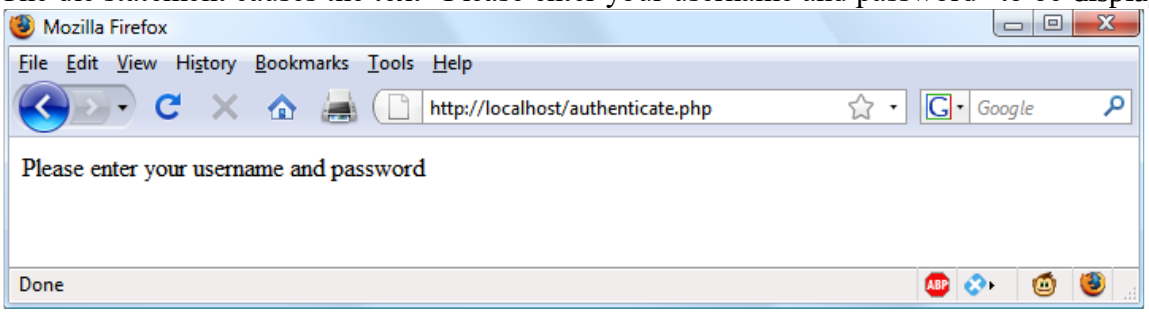
Output:



- If the user fills out the fields, the PHP program runs again from the top. But if the user clicks the Cancel button, the program proceeds to the following two lines, which send the following header and an error message:

  HTTP/1.0 401 Unauthorized

The die statement causes the text "Please enter your username and password" to be displayed:



❖ Registering Session variables:

- A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.
- Tip: If you need a permanent storage, you may want to store the data in a database.
- Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).
- Starting a PHP session:
  - A PHP session is easily started by making a call to the session_start() function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page.
  - Session variables are stored in associative array called $_SESSION[]. These variables can be accessed during lifetime of a session.
  - Example 1:

```
<?php
session_start();
?>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

Output:

Session variables are set.

- Another way to show all the session variable values for a user session is to run the following code:

```
<?php
session_start();
?>
<html>
<body>
<?php
/*// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";*/
print_r($_SESSION);
?>
</body>
</html>
```

Output:

Array ([favcolor] => green [favanimal] => cat)

- Make use of isset() function to check if session variable is already set or not.
- Example:

```
<?php
session_start();
if( isset( $_SESSION['counter'] ) ) {
 $_SESSION['counter'] += 1;}
else {
    $_SESSION['counter'] = 1; }
$msg = "You have visited this page ".  $_SESSION['counter'];
$msg .= "in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
<body>
 <?php  echo ( $msg ); ?>
 </body>
</html>
```

Output:

You have visited this page 1in this session.

❖ Destroying the variables and Session:

- A PHP session can be destroyed by session_destroy() function. This function does not need any argument and a single call can destroy all the session variables.
- If you want to destroy a single session variable then you can use unset() function to unset a session variable.
- To remove all global session variables and destroy the session, use session_unset() and session_destroy().

- Example :

```
<?php
session_start();
?>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
echo"All session variables are now removed and the session is destroyed.";
?>
```

Output:

All session variables are now removed and the session is destroyed.

INSTITUTE OF MANAGEMENT AND INFORMATION TECHNOLOGY

Subject- PHP & MYSQL(MCA 406E)

RACHANA BEHERA

Email-id: rachanabehera2792@gmail.com

UNIT 5

Database Connectivity with MYSQL:

❖ INTRODUCTION TO RDBMS:

- A Relational Database Management System (RDBMS) is a server that manages data for you.
- The data is structured into tables, where each table has some number of columns, each of which has a name and a type.
- Tables are grouped together into databases, so a James Bond database might have tables for movies, actors playing Bond, and villains.
- An RDBMS usually has its own user system, which controls access rights for databases (e.g., "user Fred can update database Bond").
- PHP communicates with relational databases such as MySQL and Oracle using the Structured Query Language (SQL).

❖ INTRODUCTION MySQL:

- MySQL is the most popular open-source database system. MySQL is a database.
- The data in MySQL is stored in database objects called tables.
- A table is a collection of related data entries and it consists of columns and rows.
- Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".

❖ CONNECTION WITH MySQL DATABASE:

Create a Connection to a MySQL Database:

- Before you can access data in a database, you must create a connection to the database.
- In PHP, this is done with the mysql_connect() function.
- Syntax: **mysql_connect(servername, username, password);**

Parameter	Description
Servername (Optional)	Specifies the server to connect to. Default value is "localhost:3306"
Username (Optional)	Specifies the username to log in with. Default value is the name of the user that owns the server process.
Password (Optional)	Specifies the password to log in with. Default is ""

- Example:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
}
// some code
?>
```

Closing a Connection:

The connection will be closed automatically when the script ends. To close the connection before, use the mysql_close() function:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
}
// some code
mysql_close($con);
?>
```

PHP mysql_pconnect () :

- The mysql_pconnect () function opens a persistent MySQL connection.
- This function returns the connection on success, or FALSE and an error on failure. You can hide the error output by adding a '@' in front of the function name.
- mysql_pconnect() is much like mysql_connect(), but with two major differences:
  - This function will try to find a connection that's already open, with the same host, username and password. If one is found, this will be returned instead of opening a new connection.
  - The connection will not be closed when the execution of the script ends (mysql_close() will not close connection opened by mysql_pconnect()). It will stay open for future use.
- Syntax:

mysql_pconnect(server, user, pwd, clientflag)

Parameter	Description
Server (optional)	Specifies the server to connect to (can also include a port number. e.g. "hostname:port" or a path to a local socket for the localhost). Default value is "localhost:3306".
User (optional)	Specifies the username to log in with. Default value is the name of the user that owns the server process.
Pwd (optional)	Specifies the password to log in with. Default is "".
Clientflag (optional)	Can be a combination of the following constants: MYSQL_CLIENT_SSL - Use SSL encryption. MYSQL_CLIENT_COMPRESS - Use compression protocol. MYSQL_CLIENT_IGNORE_SPACE - Allow space after function names. MYSQL_CLIENT_INTERACTIVE - Allow interactive timeout seconds of inactivity before closing the connection.

- Example:

```
<?php
$con = mysql_pconnect("localhost","mysql_user","mysql_pwd");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
 }
?>
```

- ✓ The reason for using PHP as an interface to MySQL is to format the results of SQL queries in a form visible in a web page. As long as you can log into your MySQL installation using your username and password, you can also do so from PHP.
- ✓ However, instead of using MySQL's command line to enter instructions and view output,you will create query strings that are passed to MySQL. When MySQL returns its response, it will come as a data structure that PHP can recognize instead of the formatted output you see when you work on the command line. Further PHP commands can retrieve the data and format it for the web page.
- ✓ The process of using MySQL with PHP is as follows:
  1. Connect to MySQL and select the database to use.
  2. Build a query string.
  3. Perform the query.
  4. Retrieve the results and output them to a web page.
  5. Repeat steps 2 to 4 until all desired data has been retrieved.
  6. Disconnect from MySQL.
  7. Most websites developed with PHP contain multiple program files that will require access to MySQL and will thus need the login and password details. Therefore, it's sensible to create a single file to store these and then include that file wherever it's needed.

PERFORMING BASIC DATABASE OPERATION:

- ❖ Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".

❖ Database Tables:

- ➢ A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.
- ➢ Below is an example of a table called "Persons":

FirstName	LastName	Age
Peter	Gen	45
Glenn	John	50

- ➢ The table above contains three records (one for each person) and three columns (FirstName, LastName, Age).

❖ Queries:

- ➢ A query is a question or a request. With MySQL, we can query a database for specific information and have a recordset returned.
- ➢ Look at query: **SELECT LastName FROM Persons**
- ➢ The query above selects all the data in the "LastName" column from the "Persons" table, and will return a record set like this:

LastName
Gen
John

Insert Data Into a Database Table:

- ➢ The INSERT INTO statement is used to add new records to a database table.
- ➢ Syntax: It is possible to write the INSERT INTO statement in two forms.
  - The first form doesn't specify the column names where the data will be inserted, only their values:

    INSERT INTO table_name
    VALUES (value1, value2, value3,...)

- The second form specifies both the column names and the values to be inserted:
  INSERT INTO table_name (column1, column2, column3,...)
  VALUES (value1, value2, value3,...)

- To get PHP to execute the statements above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.
- Example: In above table named "Persons", with three columns; "Firstname", "Lastname" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Peter', 'Griffin', '35')");
mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Glenn', 'Quagmire', '33')");
mysql_close($con);
?>
```

- Insert Data From a Form Into a Database:
  - Now we will create an HTML form that can be used to add new records to the "Persons" table. Here is the HTML form:

```
<html>
<body>
<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

  - When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php". The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables. Then, the mysql_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.
  - Here is the "insert.php" page:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{ die('Could not connect: ' . mysql_error()); }
mysql_select_db("my_db", $con);
$sql="INSERT INTO Persons (FirstName, LastName, Age)
VALUES('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";
if (!mysql_query($sql,$con))
{ die('Error: ' . mysql_error()); }
echo "1 record added";
mysql_close($con)?>
```

Select Data From a Database Table:

- The SELECT statement is used to select data from a database.
- Syntax: **SELECT column_name(s) FROM table_name**
- To get PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.
- Example: The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons");
while($row = mysql_fetch_array($result))
{
        echo $row['FirstName'] . " " . $row['LastName'];
        echo "<br />";
}
mysql_close($con);?>
```

- The example above stores the data returned by the mysql_query() function in the $result variable.
- Next, we use the mysql_fetch_array() function to return the first row from the record set as an array. Each call to mysql_fetch_array() returns the next row in the record set. The while loop loops through all the records in the record set. To print the value of each row, we use the PHP $row variable ($row['FirstName'] and $row['LastName']).
- The output of the code above will be:

```
Peter  Gen
Glenn  John
Peter    Griffin
Glenn  Quagmire
```

- Display the Result in an HTML Table:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{ die('Could not connect: ' . mysql_error()); }
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons");
echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";
while($row = mysql_fetch_array($result)){
        echo "<tr>";
        echo "<td>" . $row['FirstName'] . "</td>";
        echo "<td>" . $row['LastName'] . "</td>";
        echo "</tr>";}
echo "</table>";
mysql_close($con);?>
```

- The output of the code above will be:

Firstname	Lastname
Peter	Gen
Glenn	John
Glenn	Quagmire
Peter	Griffin

Update Data In a Database:

- The UPDATE statement is used to modify data in a table.
- The UPDATE statement is used to update existing records in a table.
- Syntax:

  UPDATE table_name
  SET column1=value, column2=value2,...
  WHERE some_column=some_value
- To get PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.
- Example:

FirstName	LastName	Age
Peter	Gen	45
Glenn	John	50
Peter	Griffin	35
Glenn	Quagmire	33

- The following example updates some data in the "Persons" table:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
mysql_query("UPDATE Persons SET Age = '36'
WHERE FirstName = 'Peter' AND LastName = 'Griffin'");
mysql_close($con);
?>
```

FirstName	LastName	Age
Peter	Gen	45
Glenn	John	50
Peter	Griffin	36
Glenn	Quagmire	33

Delete Data In a Database:

- The DELETE statement is used to delete records in a table.
- The DELETE FROM statement is used to delete records from a database table.
- Syntax:

  DELETE FROM table_name
  WHERE some_column = some_value
- To get PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

- Example:deletes all the records in the "Persons" table where LastName='Griffin':

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
        die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
mysql_query("DELETE FROM Persons WHERE LastName='Griffin'");
mysql_close($con);?>
```

Output:

FirstName	LastName	Age
Peter	Gen	45
Glenn	John	50
Glenn	Quagmire	33

❖ EXECUTING QUERY JOIN:

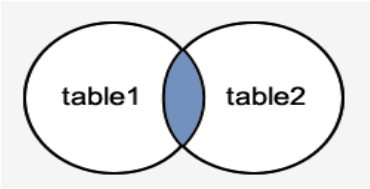
- MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.
- There are three types of MySQL joins:
  MySQL INNER JOIN (or sometimes called simple join)
  MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
  MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

MySQL Inner JOIN (Simple Join):

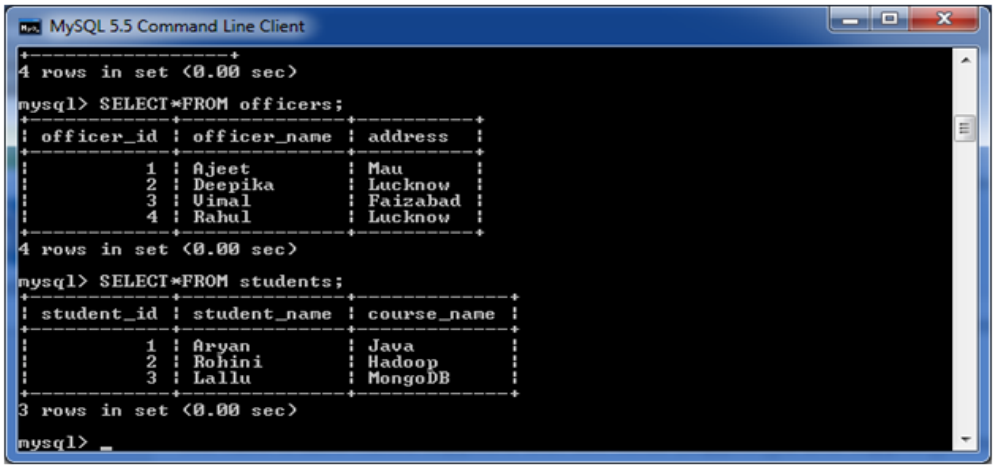
- The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.
- Syntax:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Image representation:

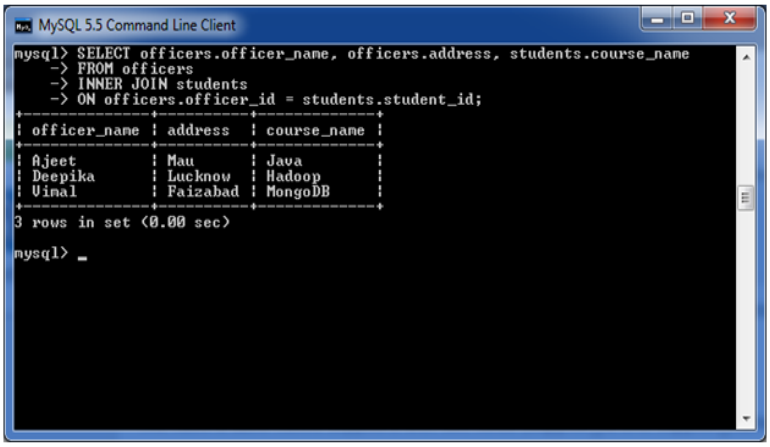


- Example: Consider two tables "officers" and "students", having the following data.



- Execute the following query:

  SELECT officers.officer_name, officers.address, students.course_name
  FROM officers
  INNER JOIN students
  ON officers.officer_id = students.student_id;
- Output:



- The MySQL Inner Join is used to returns only those results from the tables that match the specified condition and hides other rows and columns. MySQL assumes it as a default Join, so it is optional to use the Inner Join keyword with the query.
- MySQL Inner Join Example:

  Let us first create two tables "students" and "technologies" that contains the following data:

  Table: student

student_id	stud_fname	stud_lname	city
1	Devine	Putin	France
2	Michael	Clark	Australiya
3	Ethon	Miller	England
4	Mark	Strauss	America

  Table: technologies

student_id	tech_id	inst_name	technology
1	1	Java Training Inst	Java
2	2	Chroma Campus	Angular
3	3	CETPA Infotech	Big Data
4	4	Aptron	IOS

- To select records from both tables, execute the following query:

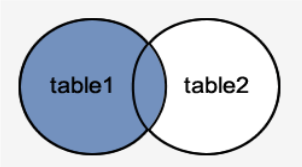
  SELECT students.stud_fname, students.stud_lname, students.city, technologies.technology
  FROM students
  INNER JOIN technologies
  ON students.student_id = technologies.tech_id;
- After successful execution of the query, it will give the following output.

stud_fname	stud_lname	city	technology
Devine	Putin	France	Java
Michael	Clark	Australiya	Angular
Ethon	Miller	England	Big Data
Mark	Strauss	America	IOS

MySQL Left Outer Join:

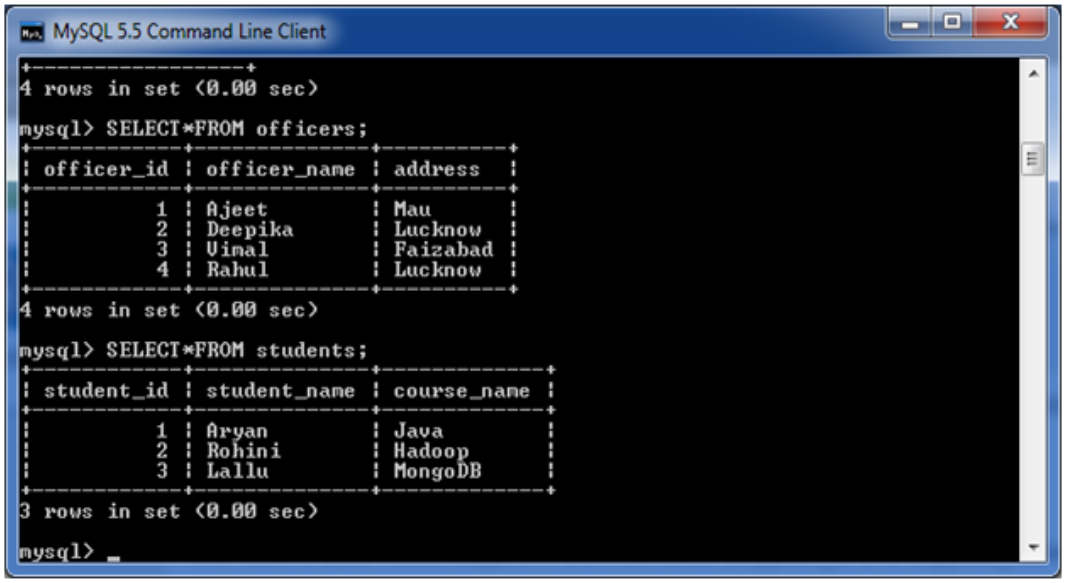
- The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.
- Syntax:

  SELECT columns
  FROM table1
  LEFT [OUTER] JOIN table2
  ON table1.column = table2.column;
- Image representation:



- Let's take an example:

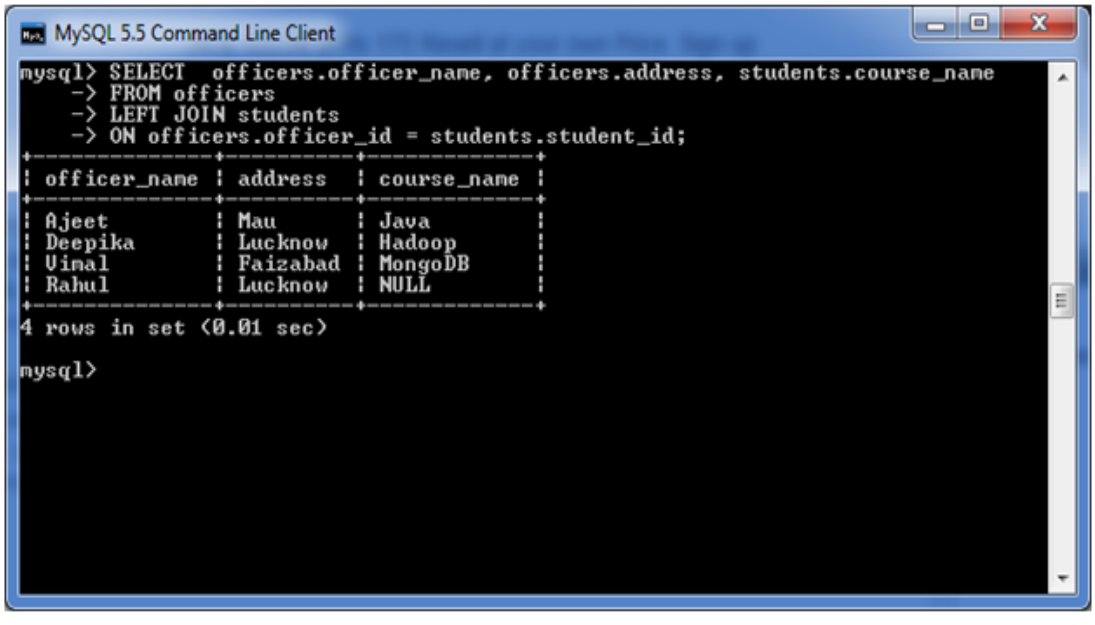
Consider two tables "officers" and "students", having the following data.



- Execute the following query:

  SELECT officers.officer_name, officers.address, students.course_name
  FROM officers
  LEFT JOIN students
  ON officers.officer_id = students.student_id;

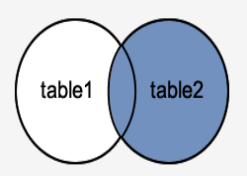
Output:



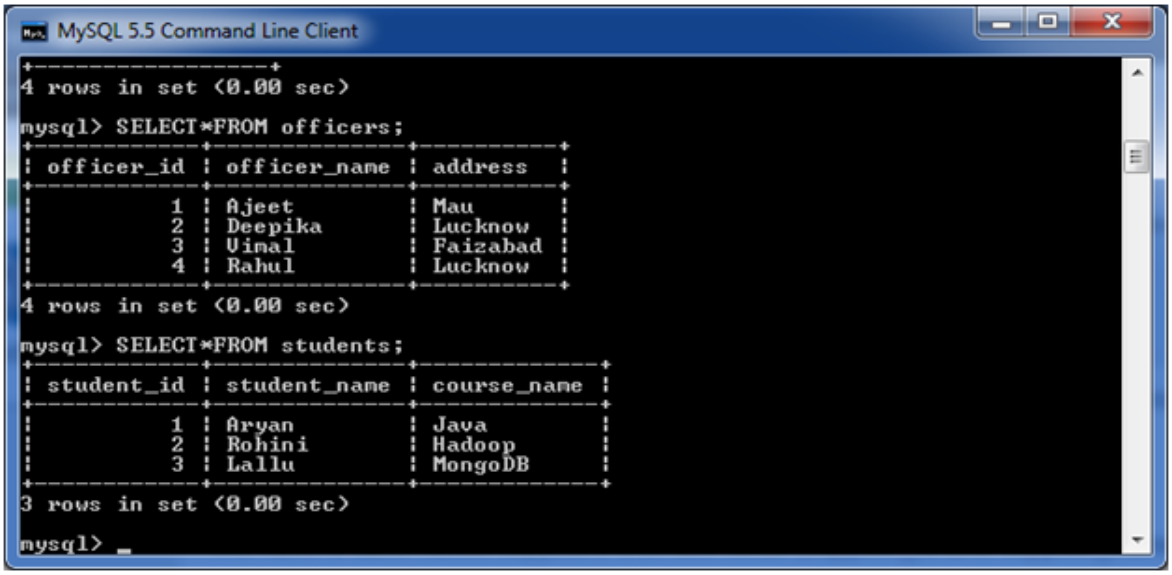
MySQL Right Outer Join:

- The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.
- Syntax:

  SELECT columns
  FROM table1
  RIGHT [OUTER] JOIN table2
  ON table1.column = table2.column;
- Image representation:



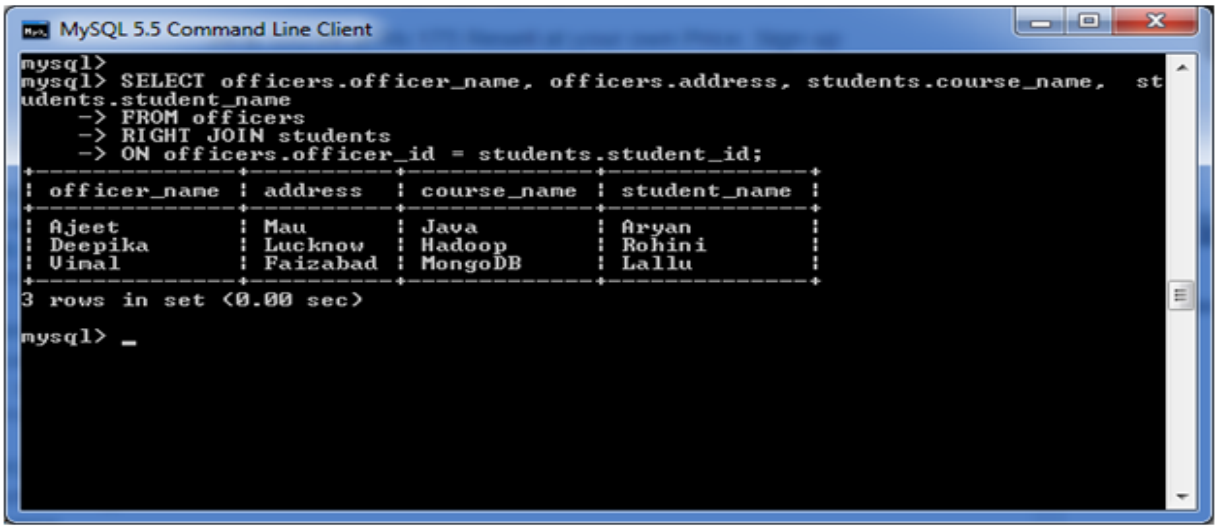
- Let's take an example:
  Consider two tables "officers" and "students", having the following data.



- Execute the following query:

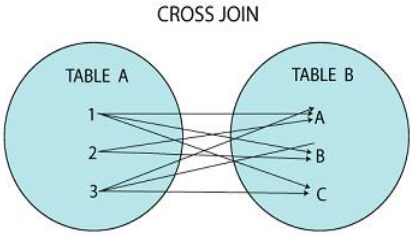
SELECT officers.officer_name, officers.address, students.course_name, students.student_name
FROM officers
RIGHT JOIN students
ON officers.officer_id = students.student_id;

- Output:



MySQL CROSS JOIN:

- MySQL CROSS JOIN is used to combine all possibilities of the two or more tables and returns the result that contains every row from all contributing tables.
- The CROSS JOIN is also known as CARTESIAN JOIN, which provides the Cartesian product of all associated tables.
- The Cartesian product can be explained as all rows present in the first table multiplied by all rows present in the second table.
- It is similar to the Inner Join, where the join condition is not available with this clause.
- MySQL CROSS JOIN Syntax:
  The CROSS JOIN keyword is always used with the SELECT statement and must be written after the FROM clause. The following syntax fetches all records from both joining tables:
  SELECT column-lists
  FROM table1
  CROSS JOIN table2;
- In the above syntax, the column-lists is the name of the column or field that you want to return and table1 and table2 is the table name from which you fetch the records.



- CROSS JOIN clause for joining two tables
  Here, we are going to create two tables "customers" and "contacts" that contains the following data:

Table: customers

customer_id	cust_name	occupation	income	qualification
1	John Miller	Developer	20000	Btech
2	Mark Robert	Enginneer	40000	Btech
3	Reyan Watson	Scientists	60000	MSc
4	Shane Trump	Businessman	10000	MBA
5	Adam Obama	Manager	80000	MBA
6	Rincky Ponting	Cricketer	200000	Btech

Table: contacts

contact_id	cellphone	homephone
1	6546645978	4565242557
2	8798634532	8652413954
3	8790744345	9874437396
4	7655654336	9934345363

- To fetch all records from both tables, execute the following query:

  SELECT *
  FROM customers
  CROSS JOIN contacts;
- After successful execution of the query, it will give the following output:

customer_id	cust_name	occupation	income	qualification	contact_id	cellphone	homephone
1	John Miller	Developer	20000	Btech	1	6546645978	4565242557
1	John Miller	Developer	20000	Btech	2	8798634532	8652413954
1	John Miller	Developer	20000	Btech	3	8790744345	9874437396
1	John Miller	Developer	20000	Btech	4	7655654336	9934345363
1	John Miller	Developer	20000	Btech	5	NULL	6786507067
1	John Miller	Developer	20000	Btech	6	NULL	9086053684
2	Mark Robert	Enginneer	40000	Btech	1	6546645978	4565242557
2	Mark Robert	Enginneer	40000	Btech	2	8798634532	8652413954
2	Mark Robert	Enginneer	40000	Btech	3	8790744345	9874437396
2	Mark Robert	Enginneer	40000	Btech	4	7655654336	9934345363
2	Mark Robert	Enginneer	40000	Btech	5	NULL	6786507067
2	Mark Robert	Enginneer	40000	Btech	6	NULL	9086053684
3	Reyan Watson	Scientists	60000	MSc	1	6546645978	4565242557
3	Reyan Watson	Scientists	60000	MSc	2	8798634532	8652413954
3	Reyan Watson	Scientists	60000	MSc	3	8790744345	9874437396

- When the CROSS JOIN statement executed, you will observe that it displays 42 rows. It means seven rows from customers table multiplies by the six rows from the contacts table.

MySQL SELF JOIN:

- A SELF JOIN is a join that is used to join a table with itself. In the previous sections, we have learned about the joining of the table with the other tables using different JOINS, such as INNER, LEFT, RIGHT, and CROSS JOIN. However, there is a need to combine data with other data in the same table itself. In that case, we use Self Join.
- We can perform Self Join using table aliases. The table aliases allow us not to use the same table name twice with a single statement. If we use the same table name more than one time in a single query without table aliases, it will throw an error.
- The table aliases enable us to use the temporary name of the table that we are going to use in the query. Let us understand the table aliases with the following explanation.
- Suppose we have a table named "student" that is going to use twice in the single query. To aliases the student table, we can write it as:

  Select … FROM student AS S1
  INNER JOIN student AS S2;
- SELF JOIN Syntax:

  The syntax of self-join is the same as the syntax of joining two different tables. Here, we use aliases name for tables because both the table name are the same. The following are the syntax of a SELF JOIN in MySQL:

  SELECT s1.col_name, s2.col_name...
  FROM table1 s1, table1 s2
  WHERE s1.common_col_name = s2.common_col_name;
- SELF JOIN Example

  Let us create a table "student" in a database that contains the following data:

student_id	name	course_id	duration
1	Adam	1	3
2	Peter	2	4
1	Adam	2	4
3	Brian	3	2
2	Shane	3	5

- Now, we are going to get all the result (student_id and name) from the table where student_id is equal, and course_id is not equal. Execute the following query to understand the working of self-join in MySQL:

```
SELECT  s1.student_id, s1.name
FROM student AS s1, student s2
WHERE s1.student_id=s2.student_id
AND s1.course_id<>s2.course_id;
```

- After the successful execution, we will get the following output:

student_id	name
1	Adam
2	Shane
1	Adam
2	Peter

PHP Exception Handling:

- Exceptions are used to change the normal flow of a script if a specified error occurs.
- With PHP 5 came a new object oriented way of dealing with errors.
- Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.
- This is what normally happens when an exception is triggered:
  - The current code state is saved
  - The code execution will switch to a predefined (custom) exception handler function
  - Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code
- We will show different error handling methods:
  - Basic use of Exceptions
  - Creating a custom exception handler
- Exceptions should only be used with error conditions, and should not be used to jump to another place in the code at a specified point.

Basic Use of Exceptions:

- When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.
- If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.
- Lets try to throw an exception without catching it:

```
<?php
//create function with an exception
function checkNum($number)
{
        if($number>1)
        {
                throw new Exception("Value must be 1 or below");
        }
        return true;
}
//trigger exception
checkNum(2);
?>
```

- The code above will get an error like this:
  Fatal error: Uncaught exception 'Exception'
  with message 'Value must be 1 or below' in C:\webfolder\test.php:6
  Stack trace: #0 C:\webfolder\test.php(12):
  checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6

Try, throw and catch:

- To avoid the error from the example above, we need to create the proper code to handle an exception.
- Proper exception code should include:
  1. **try** - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
  2. **throw** - This is how you trigger an exception. Each "throw" must have at least one "catch"
  3. **catch** - A "catch" block retrieves an exception and creates an object containing the exception information
- Lets try to trigger an exception with valid code:

```
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}
//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
}
//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}?>
```

- The code above will get an error like this:
  Message: Value must be 1 or below
- Example explained:
  The code above throws an exception and catches it:
  1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is throw
  2. The checkNum() function is called in a "try" block
  3. The exception within the checkNum() function is thrown
  4. The "catch" block retrieves the exception and creates an object ($e) containing the exception information
  5. The error message from the exception is echoed by calling $e->getMessage() from the exception object
- However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

Creating a Custom Exception Class:

- To create a custom exception handler you must create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.
- The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.
- Lets create an exception class:

```
<?php
class customException extends Exception
 {
        public function errorMessage()
        {
                //error message
                 $errorMsg='Error on line '.$this->getLine().' in '.this->getFile().':
                <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
                 return $errorMsg;
        }
}
$email = "someone@example...com";


try
{
        //check if
         if(filter_var($email,FILTER_VALIDATE_EMAIL)=== FALSE)
         {
                //throw exception if email is not valid
                throw new customException($email);
         }
}
catch (customException $e)
{
        //display custom message
        echo $e->errorMessage();
}
?>
```

- The new class is a copy of the old exception class with an addition of the errorMessage() function. Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like getLine() and getFile() and getMessage().
- Example explained:

The code above throws an exception and catches it with a custom exception class:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The $email variable is set to a string that is not a valid e-mail address
4. The "try" block is executed and an exception is thrown since the e-mail address is invalid
5. The "catch" block catches the exception and displays the error message