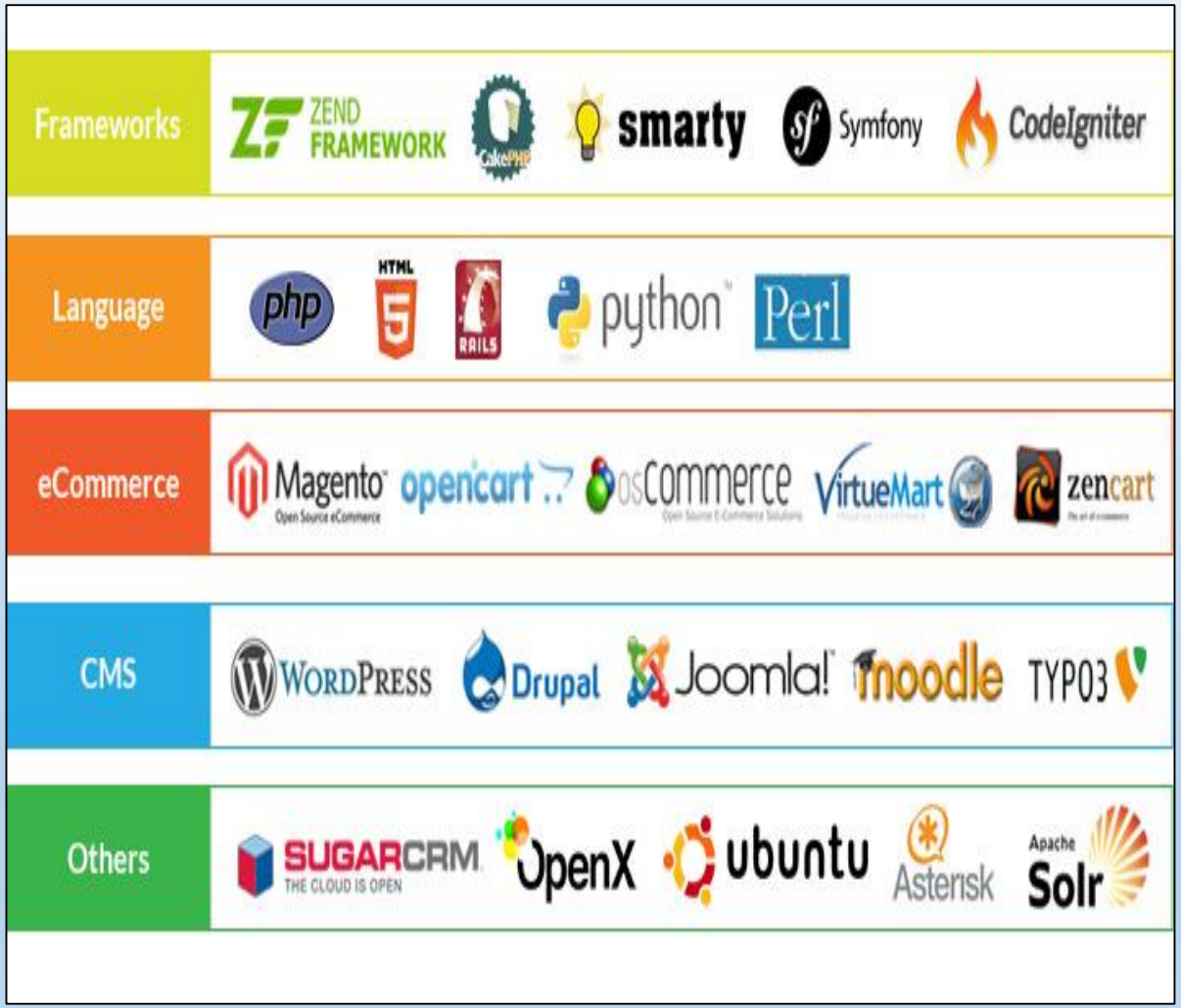
CA308: Introduction to open Source Technology

Unit 1: Introduction

Introduction to open Source Technology

- Open source software is usually a free software product, where developers have access to the source code. They can enhance the program's performance, add some features, and fix errors.
- *Open source code is written by developers and for developers.*
- The term Open-source is closely related to Open-source software (OSS). Open-source software is a type of computer software that is released under a license, but the source code is made available to all the users.
- Open-source software (OSS) is any computer software that's distributed with its source code available for modification. That means it usually includes a license for programmers to change the software in any way they choose: They can fix bugs, improve functions, or adapt the software to suit their own needs.
- Open source code is typically created as a collaborative effort in which programmers improve upon the code and share the changes within the community.

Widely used open Source Technology



Development Philosophy of open Source Technology

- According to “The Cathedral and Bazaar Model” the Open Source Software are being developed which has following pattern.
  - **Users should be treated as co-developers:**
    - Users should have access to source code
    - Users are encouraged to submit additions to s/w, code fixes for the s/w, report bugs and documentation.
    - More no. of co-developers help in building s/w faster.
    - Linus’s law says :” Given enough eyeballs all bugs are shallow.” – Many users view same source code then they find all bugs and ways to solve them.
  - **Early Releases:**
    - First version of software should be released as soon as possible to find co-developers earlier.
  - **Frequent Integration**
    - Code changes should be integrated as frequently as possible to avoid integration at the end of the project life-cycle.

Development Philosophy of open Source Technology

- **Several versions :**
  - There should be at least 2 versions of software:
    1. Development version with more no. of features.
    2. Stable version with fewer number of features.
  - Development version should be used by the developer willing to take risk of the code which is still not developed completely and want to use the latest features.
  - Such user can act as a co-developer and help in reporting bugs as well as providing fixes to those bugs.
- **High modularization:**
  - The general structure of the software should be modular allowing for parallel development on independent components.
- **Dynamic Decision-Making Structure**
  - There is a need for a decision making structure, whether formal or informal, that makes strategic decisions depending on changing user requirements and other factors.

Pros and cons of open Source Technology

- **Pros or advantages of open source technology**
  - **Cost Effective**
    - Generally an open source software is free to use. The users does not require to pay for the usage. It can be installed free of cost without any upfront expenses. Nevertheless the quality is not compromised. Especially for businesses with tight budget, using open source software can lead to significant amount of cost savings.
  - **Reliability**
    - High reliability can be ensured when using an open source software. The open source software is created by expert developers and guided by them. Therefore, there is very less chance that someone will identify flaws in the software. In addition to that, the software is monitored by a set of experts who is able to fix faults in a lesser time duration.
  - **Flexibility**
    - Unlike in proprietary software, users in open source software does not experience vendor lock-ins. Even if there are irrelevant features, the users can effectively remove them. This is because the source code is free to modify. Users do not need to stick to the provider's requirements.

Pros and cons of open Source Technology

- **Pros or advantages of open source technology**
  - **Scalability**
    - Similar to a Linux operating system, an open source software can also be scaled easily. According to the needs, an open source software can be scaled up and down so that businesses could achieve their desired results.
  - **Licensing**
    - Open source software offers convenient license schemes. Thus, the users do not need to worry about monitoring and tracking. And regardless of the location, an open source software could also be used. Moreover, it does not pose any restrictions on the number of times it can be installed.
  - **Error Free**
    - The number of people involve in maintaining an open source software is more in hundreds and thousands. Usually they are present as a community. These people make sure that the software is error free of bugs.
    - Once a user identifies a bug, they could report it to the community so that the community could fix it as soon as possible. Releasing a patch will not take longer time since the source code is available to everyone.

Pros and cons of open Source Technology

- **Cons or disadvantages of open source technology**
  - **User Friendly**
    - Not all the open source applications are easy to use. The Graphical User Interface may not be user friendly for the users to understand. Only tech savvy people will be able to understand this interface.
    - If used in an organization, the employees must be trained accordingly to operate it. For this the companies need to hire trainers all which can cause additional expenses.
  - **Security**
    - As mentioned earlier, the source code present in an open source software is free to edit. The problem comes when someone misuses the code for their own benefits. While many of them use it for improving the software, others could invite vulnerabilities such as identity theft and malicious transfers.
  - **Compatibility**
    - Some applications cannot support both open source software and proprietary software. If proprietary hardware needs to run on an open source machine, it requires third party drivers. However, that cannot guarantee that the hardware will work on the host machine. Therefore, before purchasing a hardware it is recommended to check it is supported by the open source application.

Pros and cons of open Source Technology

- **Cons or disadvantages of open source technology**
  - **Maintenance**
    - Even though an open source software does not contain any upfront costs, still it is equipped with some hidden costs. Especially with maintenance. If you face any problems in the implementation process, a third party support may be needed which could charge additionally. This is a problem for businesses that is unaware of the hidden costs.
  - **Drivers**
    - Finding required drivers for hardware components is not an easy task because the respective driver must be supporting the current operating system you use. Hence, a new hardware component that is installed could fain to provide its functionalities.
  - **Support**
    - When using an open source software, it is common to experience technical issues. Unlike a proprietary software, an open source software does not offer extensive support. For fixing this people look for support services online. Since an open source software is created by numerous developers, there is not just one person liable for an error. The only option available is to contact the provider or rely on third party support services

Open source vs. Close source software

Parameters	Open Source Software	Closed Source Software
Short-form	Also generally referred to as OSS.	Also generally referred to as CSS.
Basics	These refer to that computer software in which the source remains open. Thus, the general public is able to access it easily and use it.	These refer to that computer software in which the source code remains closed. Thus, the general public has no access to it.
Source Code	It is open source and public.	It is closed source and protected from all.
Modification of Software	Any user or organization can easily change this code since it is available as an open-source for any person to take a look at.	Only the organization or individual that created it has access to the code. Only they can modify it.
Price	It is comparatively cheaper and more cost-effective.	It is comparatively expensive and less cost-effective.
Restrictions	Users have no restrictions on the modification or usability of the software.	Users are very restricted on the modification or usability of the software.

Open source vs. Close source software

Parameters	Open Source Software	Closed Source Software
Programmer Feedback	All programmers are free to provide improvement strategies, and they get incorporated if they get accepted by the software developers.	The software organization/firm hires programmers to provide an improvement on their software.
Team Strength	A large OSS may include a large number of programmers for completing their projects in teams.	The number of programmers who might team up and work on a CSS project is always limited.
Software Responsibility	No one is explicitly responsible for an OSS.	The vendor is responsible for anything that might happen to a CSS.
Examples	Firefox, OpenOffice, Android,, MySQL, Moodle, Perl, PHP, and many more.	Google Earth, Skype, Java, Adobe Reader, Virtual Box, WinRAR, Microsoft Windows, Microsoft Office, Mac OS, and many more.

Introduction to Webpage and Website

- **Web Page**
  - web page is a document available on world wide web. Web Pages are stored on web server and can be viewed using a web browser.
  - A web page can contain huge information including text, graphics, audio, video and hyper links. These hyper links are the link to other web pages.
- **Web Site**
  - Collection of linked web pages on a web server is known as website. There is unique Uniform Resource Locator (URL) is associated with each web page.
- **Web Server**
  - A web server is a computer hosting one or more websites. "Hosting" means that all the web pages and their supporting files are available on that computer.
- **Web Browser**
  - The software is used for browsing and exhibiting web pages that are spread all across the internet.
  - It is a form of application software developed for retrieving and viewing information and content from web pages of HTML files stored in web servers.

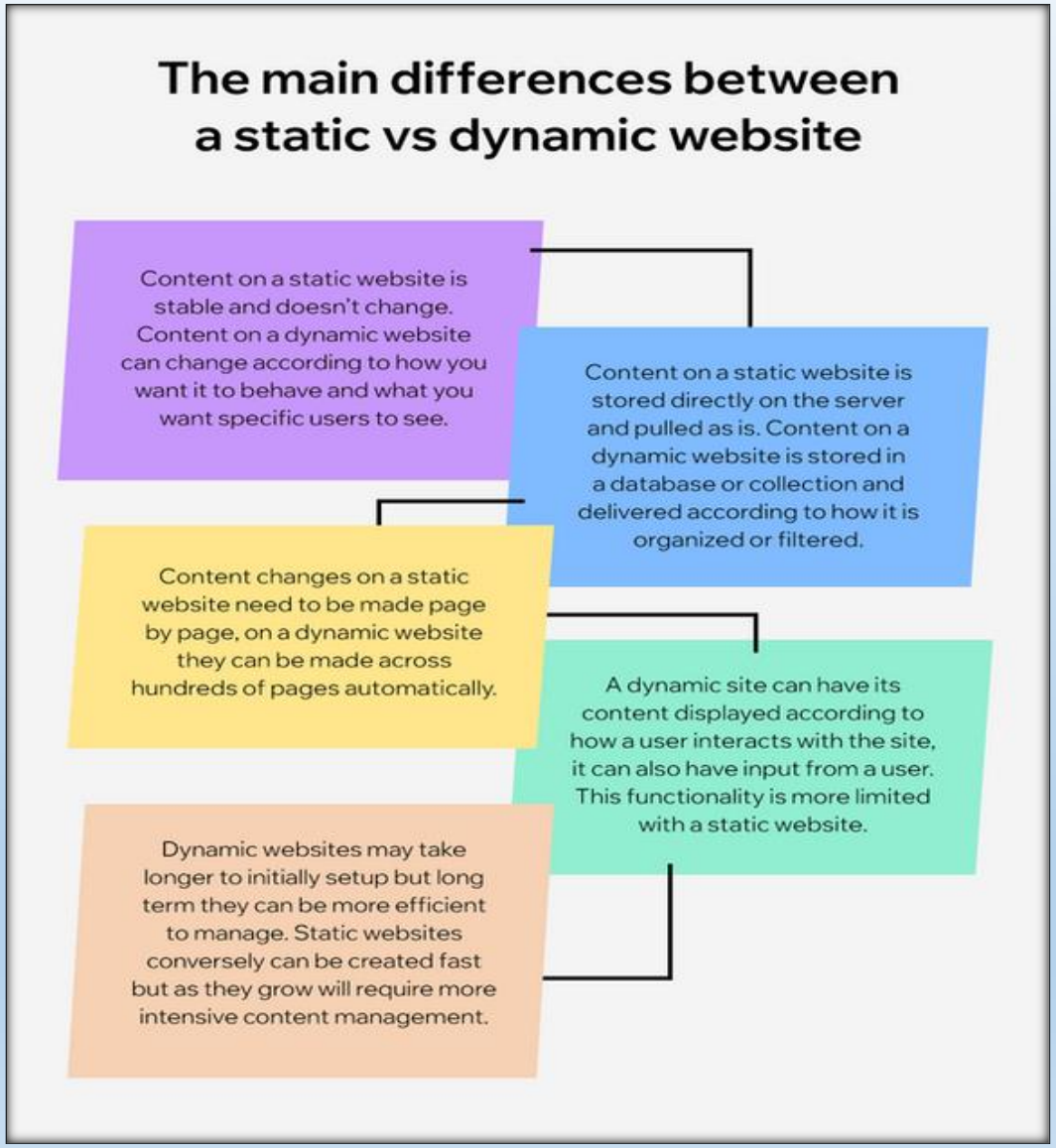
Static Website and Dynamic Website

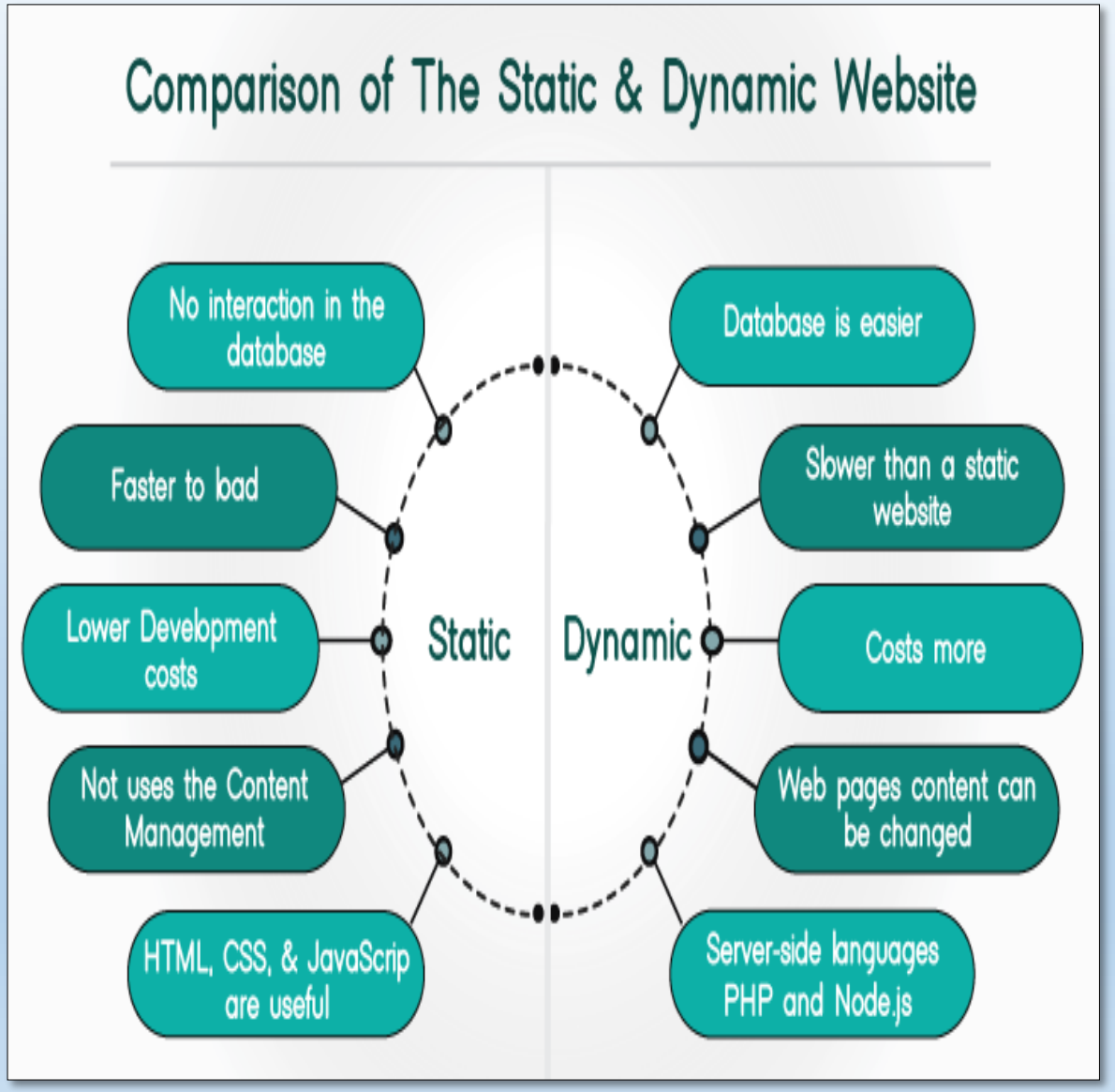
- **Static Website**
  - A static website is made up of webpages created using HTML, CSS and Javascript etc.
  - Each page on a static website is stored as a single HTML file , which is delivered directly from the server to the webpage exactly as is.
  - This content essentially becomes a part of the design on your page, and won't change unless the original HTML file is edited at a code level.
  - Changes to a static website can be done manually, and will only be made page by page, HTML file by HTML file. For example, edits made to the HTML file of a homepage will only be reflected on the homepage.
  - One of the most characteristic aspects of a static site is that every user receives and views the exact same content. Because of this, static websites work best for sites with fewer pages that don't require frequent updates or changes.

Static Website and Dynamic Website

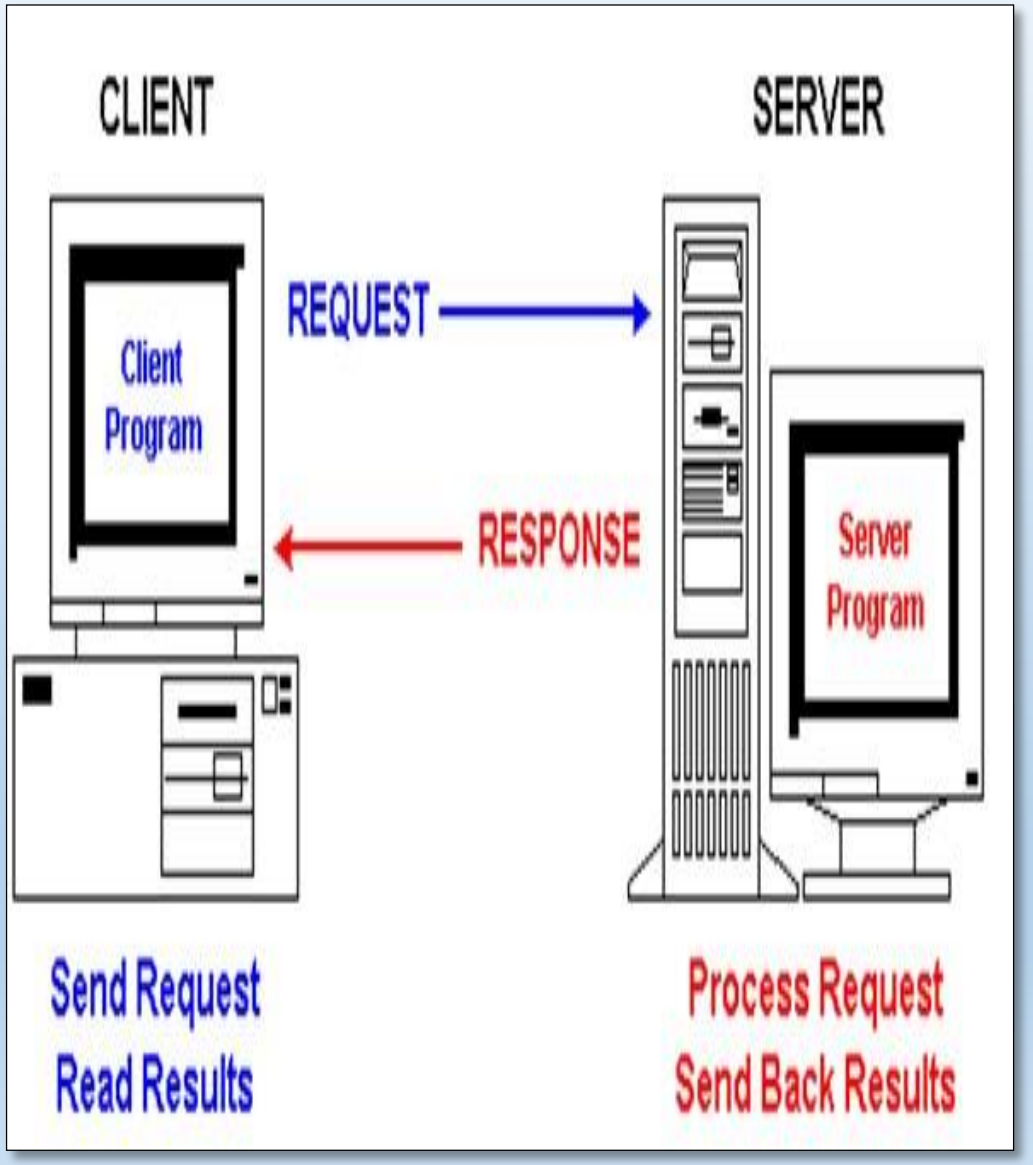
- **Dynamic Website**
  - Built using server side language and technology, dynamic websites allow for the content of each page to be delivered and displayed dynamically, or on-the-fly, according to user behavior or from user-generated content.
  - With a dynamic website all of your data and content are organized in a database or backend Content Management System (CMS), which connects to your website pages. The way this information is arranged and connected to your site's design controls how and when its content is revealed on a page.
  - Dynamic websites enable you to choose which information is displayed to a user based on their location. You can also deliver content to users based on their current or past actions on your site, which essentially means each visitor sees a different view of the content on a page.
  - Dynamic websites are those which are content heavy and user-driven.

Static Website and Dynamic Website

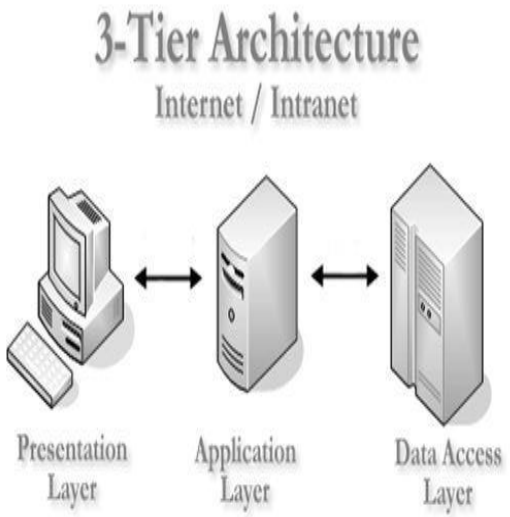


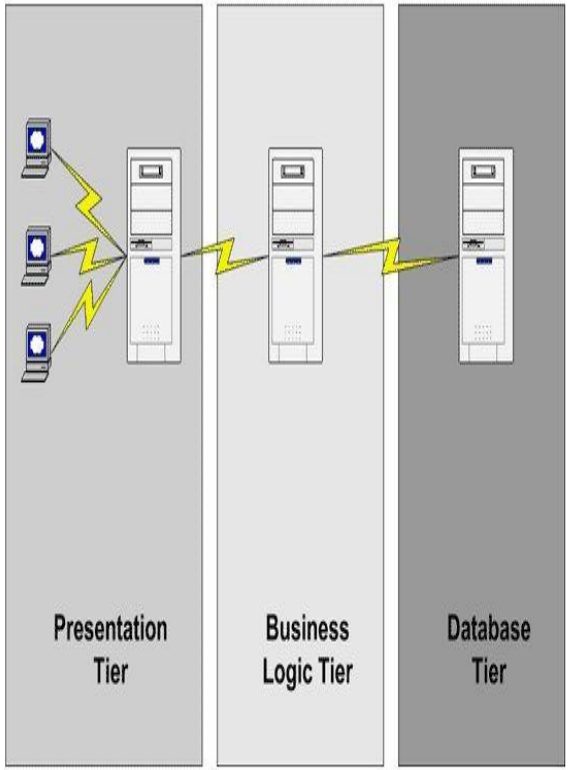


Client / Server Architecture



- In 3-tier architecture, an application is virtually split into three separate logical layers





Scripting Languages

- Scripting languages are like programming languages that allow us to write programs in form of script.
- These scripts are interpreted not compiled and executed line by line.
- Scripting language is used to create dynamic web pages.
- Scripting languages are used in web applications.
- It is used in server side as well as client side.
- Server side scripting languages are: PHP, Perl etc. and client side scripting languages are: JavaScript, AJAX, jQuery etc.

Client & Server Side Scripting

- **Client-side Scripting**
  - Client-side scripting refers to the programs that are executed on client-side.
  - Client-side scripts contains the instruction for the browser to be executed in response to certain user's action.
  - Client-side scripting programs can be embedded into HTML files or also can be kept as separate files.
- **Server-side Scripting**
  - Sever-side scripting acts as an interface for the client and also limit the user access the resources on web server.
  - It can also collects the user's characteristics in order to customize response.

Uses and example of client-side scripting

- Makes interactive web pages
- Make stuffs work dynamically
- Interact with temporary storage
- Works as an interface between user and server
- Sends requests to the server
- Retrieval of data from Server
- Interact with local storage
- Provides remote access for client server program
- There are many client-side scripting languages too. For example JavaScript, VBScript, HTML (Structure), CSS (Designing), AJAX, jQuery etc.

Uses and example of server-side scripting

- It processes the user input
- Displays the requested pages
- Structure web applications
- Interaction with servers/storages
- Interaction with databases
- Querying the database
- Encoding of data into HTML
- Operations over databases like delete, update.
- There are several languages that can be used for server-side programming: for example PHP, ASP.NET (C# OR Visual Basic), C++, Java and JSP, Python, Ruby on Rails and so on.

Other server side languages

- Server-side code can be written in any number of programming languages — examples of popular server-side web languages include PHP, Python, Ruby, C#, and JavaScript (NodeJS).

Learning is never done without errors...!!!

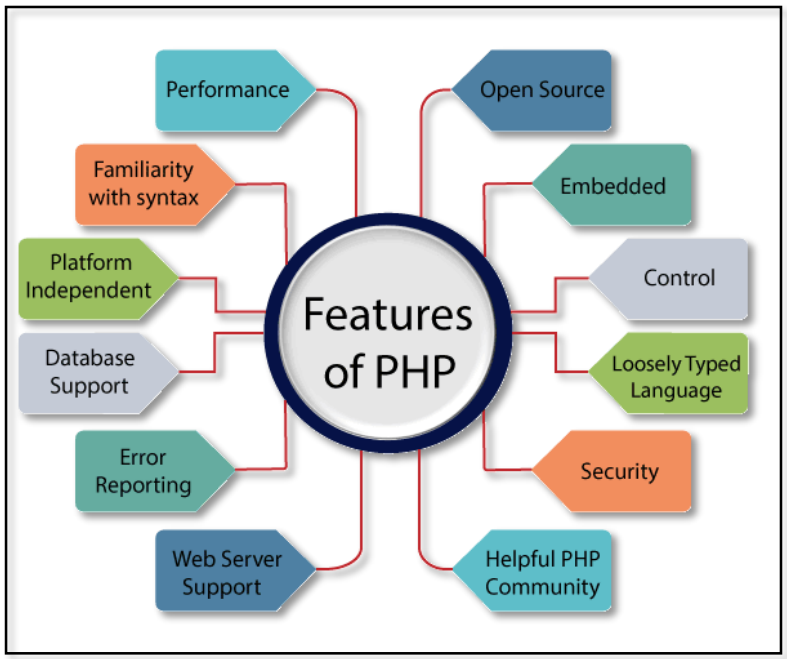
Unit-II PHP Configurations and Basic

Introduction to PHP

- PHP is an acronym for "PHP: Hypertext Pre-processor or Personal Home Page"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use.
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ". php"

PHP Features

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



- Performance:
  - PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.
- Open Source:
  - PHP source code and software are freely available on the web.

- You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

- Familiarity with syntax:
  - PHP has easily understandable syntax. Programmers are comfortable coding with it.
- Embedded:
  - PHP code can be easily embedded within HTML tags and script.
- Platform Independent:
  - PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- Database Support:
  - PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
- Error Reporting :
  - PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.
- Loosely Typed Language:
  - PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.
- Web servers Support:
  - PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
- Security:
  - PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
- Control:
  - Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.
- A Helpful PHP Community:
  - It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

Installation of Apache, MySQL and PHP

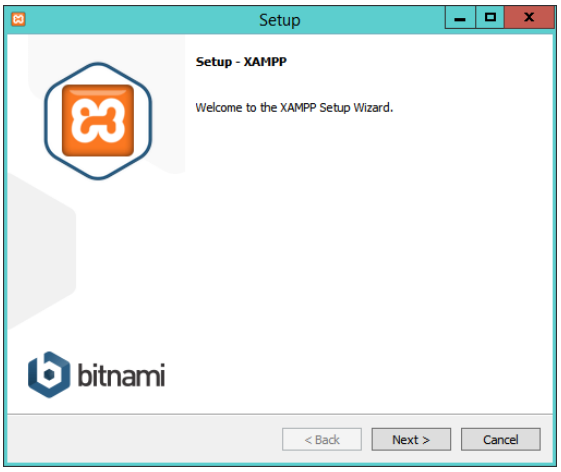
- If you wish to develop Internet applications but don't have your own development server, you can develop one such on your local computer.
- So, testing can be as easy as saving an update (usually just a matter of clicking once on an icon) and then hitting the Refresh button in your browser.
- Following are the fully functioning setup used for developing dynamic Internet web pages.
    1. WAMP - Windows, Apache, MySQL, and PHP
    2. MAMP - Mac, Apache, MySQL, and PHP
    3. LAMP - Linux, Apache, MySQL, and PHP
- WAMPs, MAMPs, and LAMPs come in the form of a package that binds the bundled programs together so that you don't have to install and set them up separately.
- This means you can simply download and install a single program, and follow a few easy prompts, to get your web development server up and running in the quickest time with a minimum hassle.

Installing Xampp on windows

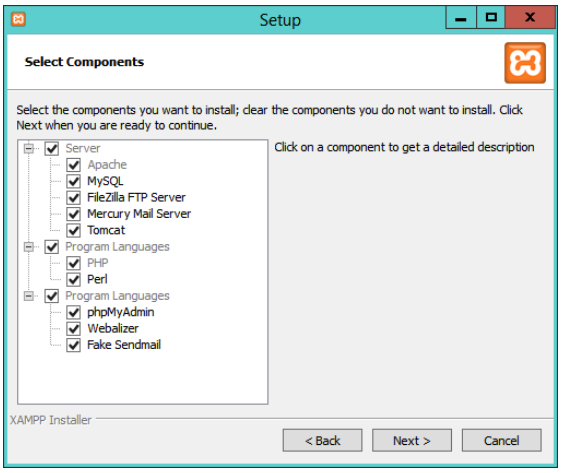
- There are several available WAMP servers, each offering slightly different configurations, but out of the various open source and free options, the best is probably XAMPP.

Link: *http://apachefriends.org*

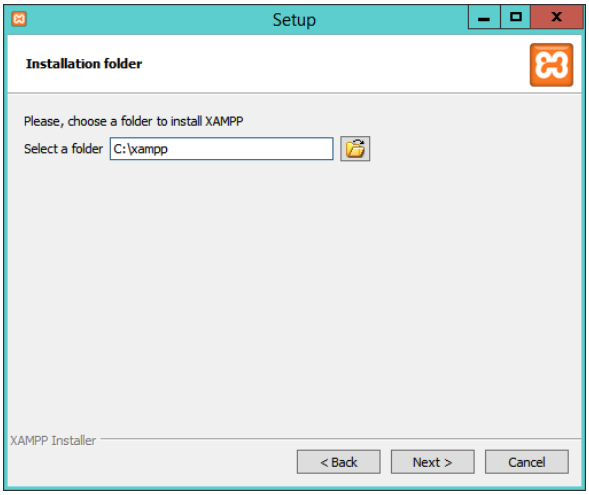
Installation Steps



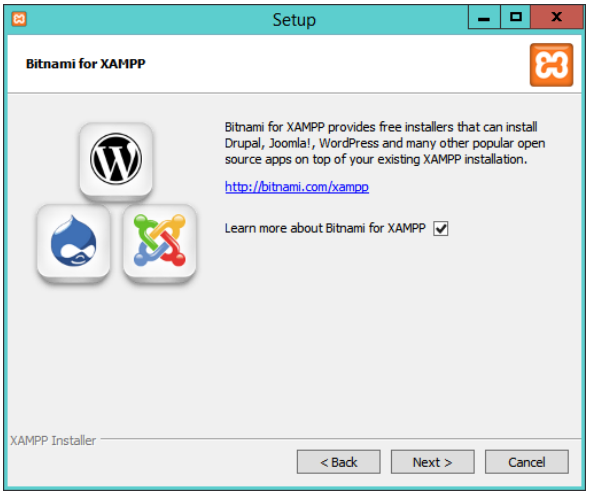
Once you've downloaded the installer, run it. It will show the following window.



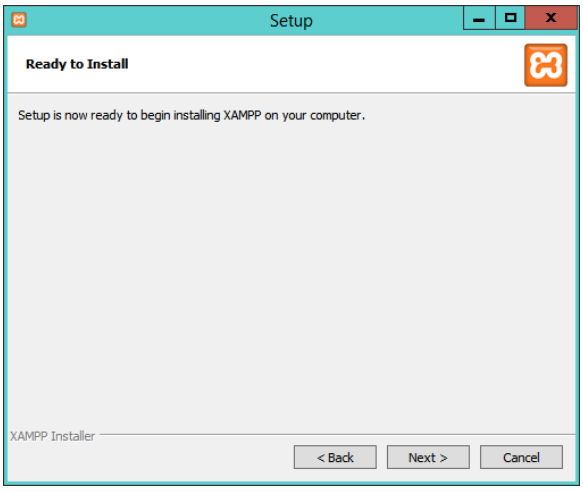
Uncheck any of the components that you don't need. For example, at the minimum for this book you will want to keep Apache, MySQL, PHP, and PHPMyAdmin checked.



Choose a folder for the installation. You are recommended to accept the default provided unless you have a good reason to choose a different one. If the folder you choose exists and is not empty, you will be unable to use it.

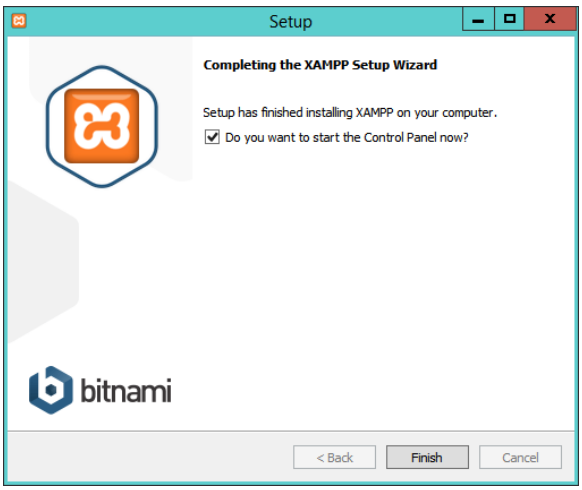


The screen is for supplying information on adding free installers for related products in a new web browser window or tab.

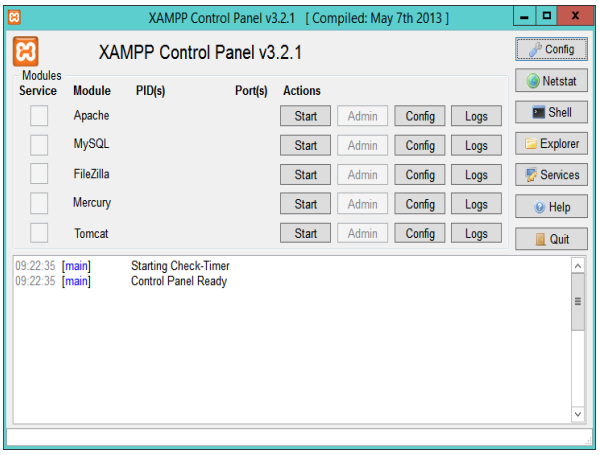


Setup is ready to start. Click next

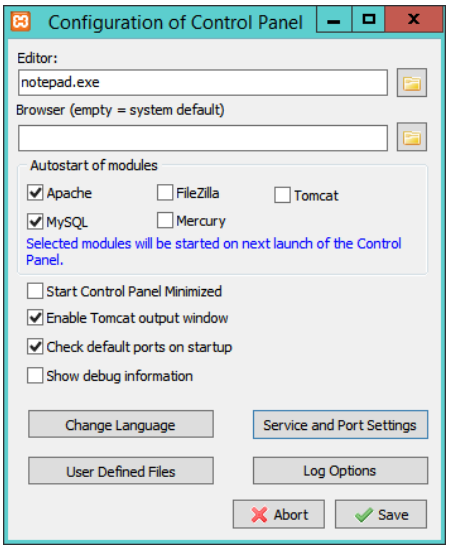
Installation screen. During installation, you can click any of the icons to open up a web browser window displaying information on the product shown. The whole process should take only a few minutes on most computers.



Once installation is complete, the screen will be displayed with a check box already checked for starting the XAMPP control panel. Click Finish.



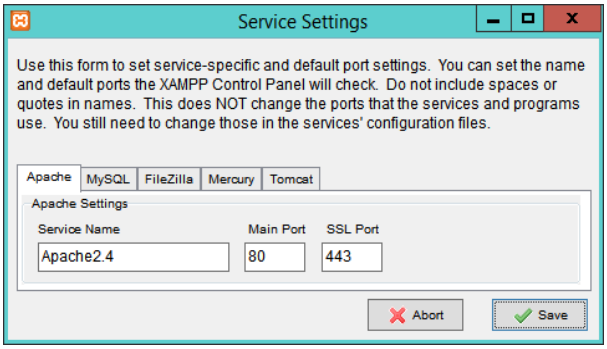
Now you are ready to start using XAMPP by configuring it from the control panel. Click the Config button at the top-right corner



Check the Apache and MySQL boxes to ensure they auto start.

OR

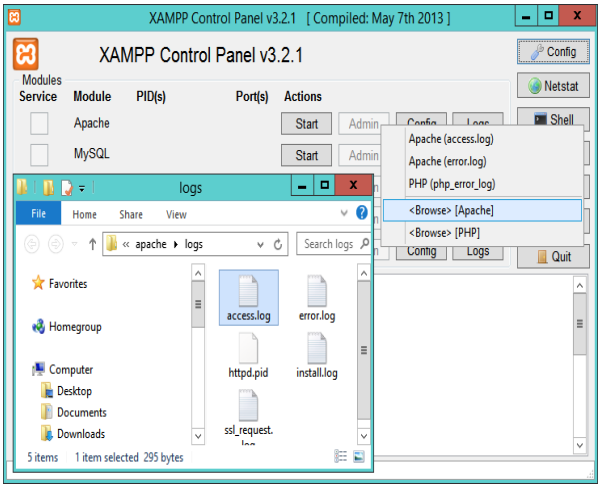
You can simply click the Start buttons for Apache and MySQL to have them run for this session only.



Alter the port settings used by clicking Service and Port Settings to call up the window

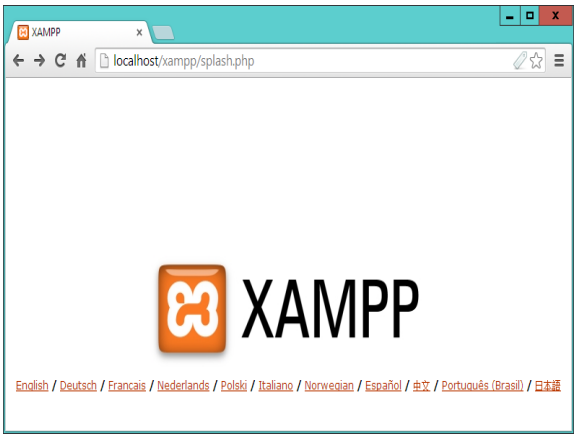
The default ports assigned will normally be port 80 for the Apache web server, 443 for

SSL, and 3306 for MySQL.



If the Apache Log button has been clicked, and the folder of log files

opened.



Browse in your default browser

Localhost Or 127.0.0.1

Port number if you changed

localhost:<port>

127.0.0.1:<port>

How PHP Code is parsed?

- So you have a file, and in that file you have some HTML and some PHP code. This is how it all works, assuming a PHP document with an extension of ". php".
- The web browser requests a document with ". php" extension.
- The web server says, "Hey! Someone wants a PHP file, which means this is a file that needs to be parsed," and sends the request on to the PHP parser.
- The PHP parser finds the requested file and scans it for PHP code.
- When the PHP parser finds PHP code, it executes that code and places the resulting output (if any) into the place in the file formerly occupied by the code.
- This new output file is sent back to the web server.
- The web server sends it along to the web browser.
- The web browser displays the output.

Embedding PHP Code and HTML?

- You can incorporate this into an HTML document by simply adding HTML outside the PHP start tag <?php and end tag ?>.
- When PHP parses a file, it looks for opening and closing tags, which tell PHP to start and stop interpreting the code between them.
- Parsing in this manner allows PHP to be embedded in different documents, as the PHP parser ignores everything outside of a pair of PHP opening and closing tags.
- If you were to view the document source, it would look exactly like a normal HTML document.
- A PHP code block can be placed anywhere in an HTML document.
- Although you can have multiple blocks of code in a single document, they combine to form a single script.

- Any variables defined in the first block will usually be available to subsequent blocks.

Example

```
<?php $var = "Hello Good Morning students."; ?>
<html>
<body>
        <h1>Multiple PHP blocks in a document</h1><hr>
        <?php echo "<h3>Message from php variable</h3>"; ?>
        <h2 align="center">
                <?php echo $var; ?>
        </h2>
</body>
</html>
```

Data Types

- Some programming languages demand that the programmer declares in advance which type of data a variable will contain.
- PHP is **loosely typed**, meaning that it will calculate data types as data is assigned to each variable.
- **Advantage:** The variable can be used flexibly, holding a string at one point and an integer at another.
- **Disadvantage:** This can lead to problems in larger scripts if you expect a variable to hold one data type when in fact it holds something completely different.

Standard Data Types

Type	Example	Description
Integer	5	A whole number
Double	3.234	A floating-point number
String	“hello”	A collection of characters
Boolean	True	One of the special values true or false
Object	$con=new DBconnection();	An instance of a class
Array	$arr=array(“rno”=>21,”name”=> ”Ram”);	An ordered set of keys and values

Special Data Types

Type	Description
Resource	Reference to a third-party resource (a database, for example)
NULL	An uninitialized variable

- Resource types are often returned by functions that deal with external applications or files.
- The type NULL is reserved (default) for variables that have not been initialized.

Operators

- Operators are symbols that make it possible to use one or more values to produce a new value.
- A value that is operated on by an operator is referred to as an operand.
- For example – "5 + 2"; "5" and "2" are operands and are operated by the operator "+" to produce new value "7"
- The combination of operands with an operator to produce a result is called an expression.
- An expression is any combination of functions, values, and operators that resolve to a value.
- There are three types of operators.
- Firstly, there is the unary operator which operates on only one value, for example (the negation operator) or ++ (the increment operator).
- The second group is termed binary operators; this group contains most of the operators that php supports. For example, logical, conditional operator etc…
- The third group is the ternary operator ?: it should be used to select between two expressions depending on a third one.

Assignment operators

Operator	Example	Is equivalent to	Description
=	$x=$y	$x=$y	Assignment
+=	$x += $y	$x = $x + $y	Combined Assignment
-=	$x -= $y	$x= $x - $y	Combined Assignment
*=	$x *= $y	$x=$x * $y	Combined Assignment
/=	$x /= $y	$x=$x / $y	Combined Assignment
%=	$x%=$y	$x=$x % $y	Combined Assignment
.=	$x .= "test"	$x = $x . "test"	Combined Concatenation (string)

Arithmetic operators

Example	Name	Result
-$a	Negation	Opposite of $a. (Unary)
$a + $b	Addition	Sum of $a and $b.
$a - $b	Subtraction	Difference of $a and $b.
$a * $b	Multiplication	Product of $a and $b.
$a / $b	Division	Quotient of $a and $b.
$a % $b	Modulus	Remainder of $a divided by $b.

Increment/decrement operators

Example	Name	Effect
++$a	Pre-increment	Increments $a by one, then returns $a.
$a++	Post-increment	Returns $a, then increments $a by one.
--$a	Pre-decrement	Decrements $a by one, then returns $a.
$a--	Post-decrement	Returns $a, then decrements $a by one.

Comparison or relational operators

Example	Name	Result
$a == $b	Equal	True if $a is equal to $b otherwise false.
$a=== $b	Identical	True if $a is equal to $b, and they are of the same type. (introduced in php 4) otherwise false.
$a != $b	Not equal	True if $a is not equal to $b otherwise false.
$a <> $b	Not equal	True if $a is not equal to $b otherwise false.
$a !== $b	Not identical	True if $a is not equal to $b, or they are not of the same type. (introduced in php 4) otherwise false.
$a < $b	Less than	True if $a is strictly less than $b otherwise false.
$a > $b	Greater than	True if $a is strictly greater than $b otherwise false.
$a <= $b	Less than or equal to	True if $a is less than or equal to $b otherwise false.
$a >= $b	Greater than or equal to	True if $a is greater than or equal to $b otherwise false.

Logical operators

Example	Name	Result
$a and $b	and	True if both $a and $b are true otherwise false.
$a or $b	or	True if either $a or $b is true otherwise false.
$a xor $b	xor	True if either $a or $b is true, but not both.

! $a	not	True if $a is not true. (Unary) otherwise false
$a && $b	and	True if both $a and $b are true otherwise false.
$a || $b	or	True if either $a or $b is true otherwise false.

Ternary (conditional) operator

- The ? : or ternary operator is similar to the if statement, but returns a value derived from one of two expressions separated by a colon.
- This construct will provide you with three parts of the whole, hence the name ternary.
- The expression used to generate the returned value depends on the result of a test expression.

Syntax

- (expression) ? returned_if_expression_is_true : returned_if_expression_is_false;
- If the test expression evaluates to true, the result of the second expression is returned, otherwise, the value of the third expression is returned.

Example

```
<? php
        $a = 30;
        $b = 20;
        $x = ($a>$b) ? $a : $b;
        echo "Max = $x";
?>
```

String operators

- There are two string operators.
- The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments.
- The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side.

Example

```
<? php
    $a = "hello ";
    $b = $a . "world!";      // now $b contains "hello world!"
    $a = "hello ";
    $a .= "world!";          // now $a contains "hello world!"
?>
```

PHP Variable

- A variable is a special container, to hold a value.
- Such value can be a number, a string, an object, an array, or a Boolean.
- Variable is a fundamental unit of any programming language.
- It is used as a placeholder which occupies specific amount of memory to store and access a value.

RULES FOR DEFINING VARIABLE IN PHP

- Any name preceded by a dollar sign ($) followed by a letter or the underscore character
- A variable name cannot start with a number
- Can include letters, numbers, and the underscore.
- Cannot include spaces or special symbols.
- Variable names are case-sensitive ($age and $AGE are two different variables)

Example

```
$a, $b;                 // Variable declaration
$a = 10;                // Assignment
$rollno = 21;           // Initialization of variable
$b = 2 * sum($a,20);    // Assignment
```

Variable Scopes

- 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
    - Global variable
    - Static variable

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.

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.

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

```
<?php
        $gvar = 10;
        function scope()
        {
                static $svar=0;
                global $gvar;
                echo "<h4> Global Variable value = $gvar </h4>";
                $lvar = 100;
                echo "<h4> Local variable value = $lvar </h4>";
                $svar++;
                echo "<h4> static variable value = $svar </h4>";
        }
        scope();
        scope();
        scope();
        echo "<h4> Outside Global Variable value = $gvar </h4>";
?>
```

Comments in PHP

- A comment is text in a script that is ignored by the PHP engine. Comments can be used to make code more readable or to annotate a script.

Single-line comment

- Begin with two forward slashes (//) (C style)
- A single hash sign (#) (Shell style)
- The PHP engine ignores all text between these marks and either the end of the line or the PHP close tag

```
// this is a comment
# this is another comment
```

Multiline comment

- Begin with a forward slash followed by an asterisk (/*) and end with an asterisk followed by a forward slash (*/)

```
/* This is a comment
none of these lines will
be parsed by the
PHP engine.   */
```

Unit-III Control Structures

Control Structure

- Control structure allows you to control the flow of code execution in application. Generally, a program is executed sequentially, line by line, and a control structure allows to alter that flow, usually depending on certain conditions.
- Control structures are core features of the PHP language that allow script to respond differently to different inputs or situations. This could allow script to give different responses based on user input, file contents, or some other data.
- PHP supports a number of different control structures:
    - Conditional Statements
        - if
        - else
        - elseif
        - switch
    - Looping Statements
        - while
        - do-while
        - for
        - foreach

if statement

- An if statement is a way of controlling the execution of a statement that follows it (that is, a single statement or a block of code inside braces).
- The if statement evaluates an expression between parentheses. If this expression results in a true value, the statement is executed.
- Otherwise, the statement is skipped entirely. This enables scripts to make decisions based on any number of factors

Syntax

```
if (expression) {
   Statements;
}
```

- As described in the section about expressions, expression is evaluated to its Boolean value.

- If expression evaluates to true, PHP will execute statements, and if it evaluates to false - it'll ignore it.
- The following example would display a is bigger than b if $a is bigger than $b

Example

```
<?Php
        If ($a > $b) {
                echo "a is bigger than b";
        }
?>
```

if...else statement

- Often you'd want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what else is for.
- else extends the if statement to execute a statement in case the expression in the if statement evaluates to false.

Syntax

```
if (expression) {
   block-1
} else {
   block-2
}
```

- If expression evaluates to true, then block 1 is executed else block 2 is executed.
- For example, the following code would display a is bigger than b if $a is bigger than $b, and a is not bigger than b otherwise

Example

```
<?php
        If ($a > $b) {
                echo "a is bigger than b";
        } else {
                echo "a is not bigger than b";
        }
?>
```

elseif statement

- elseif, as its name suggests, is a combination of if and else.

- If the first expression does not evaluate to true, the first block of code is ignored.
- The else if clause then causes another expression to be evaluated.
- Once again, if this expression evaluates to true, the second block of code is executed.
- Otherwise, the block of code associated with the else clause is executed.
- You can include as many else if clauses as you want, and if you don't need a default action, you can omit the else clause.
- For example, the following code would display a is bigger than b, a equal to b or a is smaller than b

Example

```
<? php
        If ($a > $b) {
                echo "a is bigger than b";
        } else if ($a == $b) {
                echo "a is equal to b";
        } else {
                echo "a is smaller than b";
        }
?>
```

Switch

- The switch statement is similar to a series of if statements on the same expression.
- In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.

Syntax

```
switch (expression) {
case label1:
        code to be executed if expression = label1;
        break;
case label2:
         code to be executed if expression = label2;
        break;
default:
         code to be executed if any of the case does not execute.
}
```

Example

```
<? php
        switch ($x) {
                case 1: echo "number 1";                          break;
                case 2:  echo "number 2";                         break;
                case 3: echo "number 3";                          break;
                default: echo "number is not between 1 and 3";    break;
        }
?>
```

LOOPS

- Scripts can also decide how many times to execute a block of code. Loop statements are designed to enable you to achieve repetitive tasks.
- A loop will continue to operate until a condition is achieved, or you explicitly choose to exit the loop.

while loop

- while loops are the simplest type of loop in php.

Syntax

```
while (expression) {
   statements;
}
```

- The meaning of a while statement is simple. It tells php to execute the statement(s) repeatedly, as long as the while expression evaluates to true.
- The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time php runs the statements in the loop is one iteration).
- Sometimes, if the while expression evaluates to false from the very beginning, the nested statement(s) won't even be run once.

Example

```
<? php
        $i = 1;
        while ($i <= 5) {
           echo $i++;
        }
?>
```

do…while loop

- Do…while loops are very similar to while loops, except the truth expression is checked at the end of each iteration instead of in the beginning.
- The main difference from regular while loops is that the first iteration of a do-while loop is guaranteed to run (the truth expression is only checked at the end of the iteration), whereas it's may not necessarily run with a regular while loop (the truth expression is checked at the beginning of each iteration, if it evaluates to false right from the beginning, the loop execution would end immediately).

Syntax

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

Example

```
<? php
        $i = -5;
        do {
           echo $i;
        } while ($i > 0 );
?>
```

- The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to false ($i is not bigger than 0) and the loop execution ends.

for loop

- The variable controlling loop is initialized outside the while statement. The while statement then tested the variable in its expression. The variable was incremented within the code block.
- The for statement allows you to achieve this on a single line. This allows for more compact code and makes it less likely that you will forget to increment a counter variable, thereby creating an infinite loop

Syntax

```
for (initialization expression; test expression; modification expression) {
// code to be executed
}
```

- The expressions of the loop are separated by semicolons.
    - First expression initializes a counter variable (starting of loop)
    - Second expression is the test condition to stop execution (ending of loop)
    - Third expression increments/decrements the counter (to reach start to end)
- Consider the following examples. All of them display numbers from 1 to 10

```
<? php
    /* example 1 */
    for ($i = 1; $i <= 10; $i++) {
      echo $i;
    }
    /* example 2 */
    for ($i = 1; ; $i++) {
      if ($i > 10)
            break;
      echo $i;
    }
    /* example 3 */
    $i = 1;
    for (; ; ) {
      if ($i > 10) {  break;   }
      echo $i;
      $i++;
    }
    /* example 4 */
    for ($i = 1; $i <= 10; print $i, $i++);
    ?>
```

- Of course, the first example appears to be the nicest one (or perhaps the fourth), but you may find that being able to use empty expressions in for loops comes in handy in many occasions.

foreach loop

- Loops over the array given by the parameter.
- On each loop, the value of the current element is assigned to $value and the array pointer is advanced by one - so on the next loop, you'll be looking at the next element.

Syntax

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

Example

```
<?php
        $arr = array(1, 2, 3, 4);
        foreach ($arr as $value) {
           echo $value. “<br>”;
}
```

break

- Break ends execution of the current for, foreach, while, do-while or switch structure.

Example

```
<?php
        $i=1;
        while(1) {
        switch($i) {
                case 1: echo "<br>$i - first case";          break;
                case 2: echo "<br>$i - second case";         break;
                case 5: echo "<br>$i - breaking from while.";     break;
                default: echo "<br>$i - default case.";           break;
            }
            $i++;
        }
        echo "<br> Program over.";
?>
```

continue

- Continue is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration.

Example

```
<?php
   for($i=1;$i<=2;$i++){
```

```
        for($j=1;$j<=2;$j++){
          if($j==2) continue;
          echo "J = $j<br>";
        }
        echo "I = $i<br>";
      }
    ?>
```

exit() Function

- exit() function prints a message and exits the application. It's often used to print a different message in the event.
- Use exit() when there is not an error and have to stop the execution.

Syntax:

exit ("your message");

or

exit ();

Example:

```
<?php
   exit ("This is an exit function in php");
   echo "This will not printed because "
    . "we have executed exit function";
?>
```

```
<?php
   $a = 10;
   $b = 10.0;
   if($a == $b) {
     exit('variables are equal');
   }
   else {
     exit('variables are not equal');
   }
?>
```

die() Function

- die() is the same as exit(). A program's result will be an empty screen. Use die() when there is an error and have to stop the execution.

Syntax:

die ("your message"); or die ();

exit()	**die()**
Exits the process without exceptions	Can throw in an exception
Exit script and prints message	Prints a message and end a process

RETURN

- return statement or keyword immediately terminates the execution of a function when it is called from within that function.
- If return is called from a global scope, the script stops the execution of the current script. If the current script file was included using include () or required (), then control goes back to the calling file.

Syntax:

return [expression]

Example:

```
<?php
    function square($x) {
        return $x**2;
    }
    $num=5;
    echo "calling function with argument $num";
    $result=square($num);
    echo "function returns square of $num = $result";
?>

// This is greeting.php file
<?php

    echo "Welcome every one...!!!";
    return;
?>

// This is printmessage.php file
<?php
    include("greeting.php");
    echo "Thank you so much...!!! Have nice Day...!!!";
?>
```

ARRAYS

- You know that these variables are used to store values. They can store only one value at a time
- But arrays are special types of variables that enable you to store as many values as you want.
- An array in PHP is actually an ordered map. A map is a type that maps values to keys.

- The value is whatever value you associate with that position – a string, an integer, or whatever you want.
- PHP mainly offers two types of arrays:
  - Indexed Array
  - Associative Array

Creating arrays

- You can create an array using
  - either the array () function (convenient for multiple value)
  - or the array operator [] (for a few elements or changing single value)
- The array () function is usually used when you want to create a new array and populate it with more than one element at the same time.
- The array operator is used when you want to create a new array with a few elements, or when you want to add to an existing element.

Indexed Arrays

- Arrays are indexed, which means that each entry is made up of a key and a value.
- The key is the index position, beginning with 0.

Syntax

```
$arr_var = array(val1,val2,...);          // array() function
                OR
$arr_var[index]  = "value";               // array operator []
```

Examples

1. $rainbow = array("red", "orange", "yellow", "green", "blue", "indigo", "violet");
2.
```
$rainbow[] = "red";
$rainbow[] = "orange";
$rainbow[] = "yellow";
$rainbow[] = "green";
$rainbow[] = "blue";
$rainbow[] = "indigo";
$rainbow[] = "violet";
```
3.
```
$rainbow[0] = "red";
$rainbow[1] = "orange";
$rainbow[2] = "yellow";
$rainbow[3] = "green";
$rainbow[4] = "blue";
$rainbow[5] = "indigo";
$rainbow[6] = "violet";
```

- All the three examples create a seven-element array called $rainbow, with values starting at index position 0 and ending at index position 6.
- Note: In PHP it is not compulsory to provide ordered index. You can also have unordered index numbers for the elements of array. Moreover, you can store multiple datatypes of value in a single array.

Example

```
<?php
        $a[5] = "First element";
        $a[4] = 4.25;
        $a[-10] = true;
        echo $a[5],"<br>" ,$a[4] ,"<br>",$a[-10];
?>
```

Associative Arrays

- Whereas numerically indexed arrays use an index position as the key, while associative arrays utilize actual named keys.

Syntax

```
$arr_var = array(key1=>val1, key2=>val2,…);          // array() function
                OR
$arr_var[key]  = “value”;                            // array operator []
```

Example

```
<?php
        $arr = array("Name"=>"Rajanikant",
                     "Occupation" => "Superhero",
                     "Age" => 0.00,
                     "Speciality" => "There is nothing that Rajni kan't." );
        $arr["Salary"] = -9999999;
        echo $arr[“Speciality”]. “<br>”;
        foreach($arr as $key => $val) {
                echo "$key => $val<br>";
        }
?>
```

Multidimensional Arrays

- Multidimensional array holds other arrays. It is like array of arrays.
- If each set of key/value pairs constitutes a dimension, a multidimensional array holds more than one series of these key/value pairs.

Example

```
<?php
        $arr = array(
                     array("Name"=>"Shaktiman",
                           "Occupation" => "Superhero",
```

```
                    "Age" => 10,
                    "Speciality" => "Can be invisible."),
                array("Name"=>"Krish",
                    "Occupation" => "Superhero",
                    "Age" => 40,
                    "Speciality" => "Can fly." ),
                array("Name"=>"Rajanikant",
                    "Occupation" => "Superhero",
                    "Age" => 0.00,
                    "Speciality" => "There is nothing that Rajni kan't." )
                );
    echo $arr[2];           //array & notice
    echo $arr[1]['Name'];   //Krish
    foreach($arr as $element){
      foreach($element as $val){
        echo "<br>$val";
      }
      echo "<br>";
    }
?>
```

Unit-IV Functions in PHP

Functions:

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

Why use Functions?

- **Better code organization** – PHP functions allow us to group blocks of related code that perform a specific task together.
- **Reusability** – once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
- **Easy maintenance**- updates to the system only need to be made in one place.

Type of the Functions?

- Functions are of two types: (1) Built-in functions (2) User Defined Functions

Built-In Functions of PHP

- Built-in functions are the functions that are provided by PHP to make programming more convenient.
- Many built-in functions are present in PHP which can be used in our code. These functions are very helpful in achieving programming goals.
- The built-in functions in PHP are very simple and easy to use. They make PHP powerful and useful.
- PHP is very rich in terms of Built-in functions.

Array Function

Function Name	Description
sizeof($arr)	Returns the size of the array
is_array($arr)	It returns true if the variable is an array and returns false otherwise.
in_array($var, $arr)	To check whether a certain value is present in the array or not. It returns true if the variable is in array and returns false otherwise.
array_merge($arr1, $arr2)	Combine two different arrays into a single array

array_values($arr)	Retrieve only value from array
array_keys($arr)	Retrieve only key from array
array_pop($arr)	Removes the last element of the array
array_push($arr, $val)	Add a new element at the end of the array.
sort($arr)	Sorts the array elements in ascending order
array_flip($arr)	Interchange the keys and the values
array_reverse($arr)	Reverse the order of elements

String Function

Function Name	Description
strlen($str)	Returns the length of the string
str_word_count($str)	Returns the number of words in the string
strrev($str)	Used to reverse a string
strpos($str, $text)	Used to find the position of any text/word in a given string. string also assign index value to the characters stored in it, starting from zero.
str_replace($replacethis, $replacewith, $str)	Used to replace a part of the string with some text.
ucwords($str)	converts first letter of every word in the string to uppercase.
strtoupper($str)	To convert string to uppercase
strtolower($str)	To convert string to lowercase
str_repeat($str, $nooftime)	repeat a string a given number of times
strcmp($str1, $str2)	Used to compare two strings. The comparison is done alphabetically. If the first string is greater than second string, the result will be greater than 0, if the first string is equal to the second string, the result will be equal to 0 and if the second string is greater than the first string, then the result will be less than 0.
substr($str, $start, $length)	Find the substring from given string. Used to take out a part of the string(substring), starting from a particular position, of a particular length.
trim($str)	Used to remove extra whitespaces from beginning and the end of a string.

PDF Function

- FPDF is a PHP class which allows generating PDF files with PHP code. It is free to use.
- FPDF stands for Free PDF. It means that any kind of modification can be done in PDF files.
- The latest version requires at least PHP 5.1 and is compatible with PHP 7 and PHP 8.

Note: Download the latest version of this Class from http://www.fpdf.org/en/download.php

- FPDF has other benefits: high level functions. Here is a list of its main features:
  - Choice of measure unit, page format and margins
  - Page header and footer management
  - Automatic page break
  - Automatic line break and text justification
  - Image support (JPEG, PNG and GIF)
  - Colors
  - Links
  - TrueType, Type1 and encoding support
  - Page compression

AddPage()

- Adds a new page to the document.

Syntax: AddPage([string orientation [, mixed size [, int rotation]]])

Parameters

`orientation`

Page orientation. Possible values are (case insensitive):

- `P` or `Portrait`
- `L` or `Landscape`

The default value is the one passed to the constructor.

`size`

Page size. It can be either one of the following values (case insensitive):

- `A3`
- `A4`
- `A5`
- `Letter`
- `Legal`

or an array containing the width and the height (expressed in user unit).

The default value is the one passed to the constructor.

`rotation`

Angle by which to rotate the page. It must be a multiple of 90; positive values mean clockwise rotation. The default value is `0`.

SetFont()

- Sets the font used to print character strings.
- It is mandatory to call this method at least once before printing text or the resulting document would not be valid.

Syntax: SetFont(string family [, string style [, float size]])

Parameters

family

Family font. It can be either a name defined by AddFont() or one of the standard families (case insensitive):

- `Courier` (fixed-width)
- `Helvetica` or `Arial` (synonymous; sans serif)
- `Times` (serif)
- `Symbol` (symbolic)
- `ZapfDingbats` (symbolic)

It is also possible to pass an empty string. In that case, the current family is kept.

style

Font style. Possible values are (case insensitive):

- empty string: regular
- `B`: bold
- `I`: italic
- `U`: underline

or any combination. The default value is regular. Bold and italic styles do not apply to `Symbol` and `ZapfDingbats`.

size

Font size in points.
The default value is the current size. If no size has been specified since the beginning of the document, the value taken is 12.

SetFontSize()

- Defines the size of the current font.

Syntax: SetFontSize(float size) where size is the font size.

Cell()

- Prints a cell (rectangular area) with optional borders, background color and character string. The upper-left corner of the cell corresponds to the current position.
- The text can be aligned or centered.
- After the call, the current position moves to the right or to the next line. It is possible to put a link on the text.
- If automatic page breaking is enabled and the cell goes beyond the limit, a page break is done before outputting.

Syntax: Cell(float w [, float h [, string txt [, mixed border [, int ln [, string align [, boolean fill [, mixed link]]]]]]])

Parameters

`w`

Cell width. If `0`, the cell extends up to the right margin.

`h`

Cell height. Default value: `0`.

`txt`

String to print. Default value: empty string.

`border`

Indicates if borders must be drawn around the cell. The value can be either a number:

- `0`: no border
- `1`: frame

or a string containing some or all of the following characters (in any order):

- `L`: left
- `T`: top
- `R`: right
- `B`: bottom

Default value: `0`.

`ln`

Indicates where the current position should go after the call. Possible values are:

- `0`: to the right
- `1`: to the beginning of the next line
- `2`: below

Putting `1` is equivalent to putting `0` and calling Ln() just after. Default value: `0`.

`align`

Allows to center or align the text. Possible values are:

- `L` or empty string: left align (default value)
- `C`: center
- `R`: right align

`fill`

Indicates if the cell background must be painted (`true`) or transparent (`false`). Default value: `false`.

`link`

URL or identifier returned by AddLink().

Output()

- Send the document to a given destination: browser, file or string. In the case of a browser, the PDF viewer may be used or a download may be forced.

Syntax: Output([string dest [, string name [, boolean isUTF8]]])

Parameters

`dest`

Destination where to send the document. It can be one of the following:

- `I`: send the file inline to the browser. The PDF viewer is used if available.
- `D`: send to the browser and force a file download with the name given by `name`.
- `F`: save to a local file with the name given by `name` (may include a path).
- `S`: return the document as a string.

The default value is `I`.

`name`

The name of the file. It is ignored in case of destination `S`.
The default value is `doc.pdf`.

`isUTF8`

Indicates if `name` is encoded in ISO-8859-1 (`false`) or UTF-8 (`true`). Only used for destinations `I` and `D`.
The default value is `false`.

Ln()

- Performs a line break.

Syntax: Ln([float h])

Parameters

`h`

The height of the break.
By default, the value equals the height of the last printed cell.

SetFillColor()

- Defines the color used for all filling operations (filled rectangles and cell backgrounds).
- It can be expressed in RGB components or gray scale.
- The method can be called before the first page is created and the value is retained from page to page.

Syntax: SetFillColor(int r [, int g, int b])

Parameters

r

If g and b are given, red component; if not, indicates the gray level. Value between 0 and 255.

g

Green component (between 0 and 255).

b

Blue component (between 0 and 255).

SetTextColor()

- Defines the color used for text.
- It can be expressed in RGB components or gray scale.
- The method can be called before the first page is created and the value is retained from page to page.

Syntax: SetTextColor(int r [, int g, int b])

Parameters

r

If g and b are given, red component; if not, indicates the gray level. Value between 0 and 255.

g

Green component (between 0 and 255).

b

Blue component (between 0 and 255).

Example:

```php
<?php
    require ('./fpdf/fpdf.php');
    $pdf = new FPDF();
    $pdf->AddPage();
    $pdf->SetFont("Times",'B',16);

    $pdf->Cell(175,10,"CMPICA - CHARUSAT",0,1,'C');
    $pdf->SetFont("Arial",'',12);

    $pdf->setFillColor(152,222,235);
    $pdf->Cell(15,7,"1",1,0,"C");
    $pdf->Cell(20,7,"BCA",1,0,'C',true);

    $pdf->ln();
    $pdf->setTextColor(70,100,175);
    $pdf->Cell(15,7,"2",1,0,"C");
    $pdf->Cell(20,7,"MCA",1,0,'C');

    $pdf->output();
?>
```

File Handling Function

Function Syntax	Use	Parameter	Returns
bool **file_exists** (string $filename);	Checks whether a file or directory exists	Path of file or directory	TRUE if the file or directory specified by filename exists FALSE otherwise
bool **is_file** (string $filename);	Tells whether the filename is a regular file	Path to the file	TRUE if the filename exists and is a regular file. FALSE otherwise
bool **is_dir** (string $filename);	Tells whether the filename is a directory	Path to the directory	TRUE if the filename exists and is a directory FALSE otherwise
bool **is_readable** (string $filename)	Tells whether a file exists and is readable	Path to the file	TRUE if the file or directory specified by filename exists and is readable FALSE otherwise
bool **is_writable** (string $filename)	Tells whether the filename is writable	The filename being checked	Returns TRUE if the filename exists and is writable.
bool **is_executable** (string $filename)	Tells whether the filename is executable	Path to the file	TRUE if the filename exists and is executable, or FALSE on error
int **filesize** (string $filename)	Gets the size for the given file.	Path to the file	Size of the file in bytes or FALSE and generates an error of level E_WARNING in case of an error
int **fileatime** (string $filename)	Gets the last access time of the given file.	Path to the file	Time (as a Unix timestamp) the file was last accessed, or FALSE on failure.
int **filemtime** (string $filename)	Gets file modification time	Path to the file	Time (as a Unix timestamp) the file was last modified, or FALSE on failure.
int **filectime** (string $filename)	Gets last change time of file	Path to the file	Time (as a Unix timestamp) the file was last changed, or FALSE on failure.
bool touch (string $filename)	Create an empty file	Path to the file	TRUE on success, FALSE on fail
bool unlink (string $filename)	Remove file	Path to the file	TRUE on success, FALSE on fail

---

File Handling Function (Reading and Writing)

Function	Syntax	Description
fopen()	fopen(filename,mode)	The fopen() function is used to open file for reading and writing purpose base on mode. The first parameter is filename and second parameter is mode of file.
fread()	fread(filename,filesize)	The fread() function reads from an open file. The first parameter contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
fclose()	fclose(filename)	The fclose() function is used to close an open file. The fclose() requires the name of the file (or a variable that holds the filename) we want to close:
fgets()	fgets(filename)	The fgets() function is used to read a single line from a file. The fgets() requires the name of the file (or a variable that holds the filename) from which we want to read data.
feof()	feof(filename)	The feof() function checks if the "end-of-file" (EOF) has been reached. The feof() function is useful for looping through data of unknown length.
fgetc()	fgetc(filename)	The fgetc() function is used to read a single character from a file.
fwrite()	fwrite(filename,data)	The fwrite() function is used to write to a file. The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.
fseek()	fseek(filename, offset, whence)	This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.
file_get_contents()	file_get_contents(filename)	Reading whole file into a string.
file_put_contents()	file_put_contents(filename,data)	Writing whole data inti a file.

File opening modes

mode	**Purpose**	**File Pointer**	**Other**
r	Reading only	Beginning of file	NA
r+	Reading and writing		
w	Writing only	Beginning of file	Attempts to create, if file doesn't exists Content will be truncated, if file exists
w+	Reading and writing		
a	Writing only	End of file	Attempts to create, if file doesn't exists
a+	Reading and writing		
x	Create and open for writing only	Beginning of file	Attempts to create, if file doesn't exists Generate an error of level E_WARNING, if file exists
x+	Create and open for reading and writing		

CSV File Handling Function

User Define Functions of PHP

- You can define your own functions using the **function statement**

Syntax

```
function function_name($argument1, $argument2 . . .) {
//function code here
}
```

- The name of the function follows the function statement and precedes a set of parentheses.
- If your function requires arguments, you must place comma separated variable names (function parameters) within the parentheses.
- These variables will be filled by the values passed to your function.
- Even if your function doesn't require arguments, you must nevertheless supply the parentheses.

Example

```
<?php
    function message($msg) // here $msg is formal parameter
    {
        echo "<br>" . $msg;
    }
    message("hello"); // here "Hello" actual parameter
    message ("how are you");
?>
```

Note: The naming rules for functions are similar to the naming rules for variables except one that is, unlike variable name function name doesn't start with $ sign.

RETURNING VALUES FROM USER-DEFINED FUNCTIONS

- A function can return a value using the return statement.
- The return statement stops the execution of the function and sends the value back to the calling code.

Example

```
<?php
    function add($n1,$n2){
        $ans = $n1 + $n2;
        return $ans;
    }
```

```
        echo "<br>Answer = ".add(10,20);
        echo "<br>Answer = ".add(2.50,3);
        echo "<br>Answer = ".add("3","2hello");
    ?>
```

- The return statement can return a value or nothing at all. How we arrive at a value passed by return can vary. The value can be hard-coded like return 4;
- It can be the result of an expression like return $a/$b;
- It can be the value returned by yet another function call like return another_function($an_argument);

Default Value for Arguments

- PHP gives you a nifty feature to help build flexible functions.
- Until now, we've said that some functions require one or more arguments.
- By making some arguments optional, you can render your functions a little less autocratic.

Example

```
    <?php
        function default_args($a, $b=0, $c=0) {
            echo "<br>", $a + $b + $c;
        }
        default_args(10, 20, 30);     // 60
        default_args(10, 20);         // 30
        default_args(10);             // 10
    ?>
```

Note: You must define any arguments with default value to the right of any arguments without default value. For Example

```
function make ($name = "anil", $age)      // incorrect
function make ($age, $name = "anil")      // correct
```

Passing Variable Reference

- When you pass arguments to functions, they are stored as copies in parameter variables.
- Any changes made to these variables in the body of the function are local to that function and are not reflected beyond it.

Example: Pass by Value

```
    <?php
        function addFive($num) {
            $num += 5;
        }
```

```
        $orignum = 10;
        addFive($orignum);
        echo $orignum;
    ?>
```

- We can change this behavior by creating a reference to our original variable.
- You can think of a reference as a signpost that points to a variable.
- In working with the reference, you are manipulating the value to which it points.
- You can pass an argument by reference by adding an ampersand to the argument name in the function definition.

Example: Pass by Reference

```
    <?php
        function addFive(&$num) {
            $num += 5;
        }
        $orignum = 10;
        addFive($orignum);
        echo $orignum;
    ?>
```

Unit-V Working with Data

HTML FORM element

- The HTML <form> element can contain one or more of the following form elements: (reference from: https://www.w3schools.com/html/html_form_elements.asp)
  - <input>
  - <label>
  - <select>
  - <textarea>
  - <button>

The <input> Element

- One of the most used form element is the <input> element.
- The <input> element can be displayed in several ways, depending on the type attribute.

Example:

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

<h2>The input Element</h2>

<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

The <label> Element

- The <label> element defines a label for several form elements.
- The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.
- The <label> element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.
- The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

The <select> Element

- The <select> element defines a drop-down list.

- The <option> elements defines an option that can be selected.
- By default, the first item in the drop-down list is selected.
- To define a pre-selected option, add the selected attribute to the option:
- Use the size attribute to specify the number of visible values:
- Use the multiple attribute to allow the user to select more than one value.

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

<h2>Allow Multiple Selections</h2>

<p>Use the multiple attribute to allow the user to select more than one value.</p>

<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars" size="4" multiple>
    <option value="volvo" selected>Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select><br><br>
  <input type="submit">
</form>

<p>Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.</p>

</body>
</html>
```

The <textarea> Element

- The <textarea> element defines a multi-line input field (a text area).

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

<form action="action_page.php">
  <textarea name="message" rows="10" cols="30">The cat was playing in the garden.</textarea>
  <br>
  <input type="submit">
</form>

</body>
</html>
```

The <button> Element

- The <button> element defines a clickable button:

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

<button type="button" onclick="alert('Hello World!')">Click Me!</button>

</body>
</html>
```

HTML Form Input Types

- In HTML <input type=" "> is an important element of HTML form. The "type" attribute of input element can be various types, which defines information field.
- Such as <input type="text" name="name"> gives a text box.
- Following is a list of all types of <input> element of HTML.

type=" "	Description
text	Defines a one-line text input field
password	Defines a one-line password input field
submit	Defines a submit button to submit the form to server
reset	Defines a reset button to reset all values in the form.
radio	Defines a radio button which allows select one option.
checkbox	Defines checkboxes which allow select multiple options form.
button	Defines a simple push button, which can be programmed to perform a task on an event.
file	Defines to select the file from device storage.
image	Defines a graphical submit button.

- HTML5 added new types on <input> element. Following is the list of types of elements of HTML5

type=" "	Description
color	Defines an input field with a specific color.
date	Defines an input field for selection of date.
datetime-local	Defines an input field for entering a date without time zone.
email	Defines an input field for entering an email address.
month	Defines a control with month and year, without time zone.
number	Defines an input field to enter a number.
url	Defines a field for entering URL
week	Defines a field to enter the date with week-year, without time zone.
search	Defines a single line text field for entering a search string.
tel	Defines an input field for entering the telephone number.

PHP Super Global Variables

Variable	Contains
$_GET	Variables provided to a script through the GET method.
$_POST	Variables provided to a script through the POST method.
$_COOKIE	Variables provided to a script through a cookie.
$_FILES	Variables provided to a script through file uploads.
$_SERVER	Contains information such as headers, file paths, and script locations.
$_ENV	Contains any variables provided to a script as part of the server environment.
$_REQUEST	Contains any variables provided to a script via any user input mechanism.
$_SESSION	Contains any variables that are currently registered in a session.

Passing variables between pages through GET, POST and REQUEST

$_GET

- $_GET is a super global variable which is used to collect form data after submitting an HTML form with method="get".
- $_GET can also collect data sent in the URL.

First.html

```
<html>
<head>
<title>My First HTML web page</title>
</head>
<body>
    <FORM name="form1" action="result.php" method="GET">
        Name : <input type="text" name="name"><br>
        Password : <input type="password" name="pass"><br>
        <input type="submit" name="submit1" value="Login"> <br>
        <a href="result.php?name=PHP&pass=get">Click Here</a>
    </FORM>
</body>
</html>
```

- When a user clicks on the link "Click Here" and user click on login, the parameters "name" and "pass" are sent to "result.php", and access their values in "result.php" with $_GET.
- The example below shows the code in "result.php":

result.php

```
<?php
        echo "Your Username = ". $_GET["name"]. "<br/>";
        echo " Password = ".$_GET["pass"];
?>
```

$_POST

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

First.html

```
<html>
<head>
<title>My First HTML web page</title>
</head>
<body>
    <FORM name="form1" action="result.php" method="GET">
        Name : <input type="text" name="name"><br>
        Password : <input type="password" name="pass"><br>
    </FORM>
</body>
</html>
```

result.php

```
<?php
    echo "Your Username = ". $_POST["name"]. "<br/>";
    echo " Password = ".$_POST["pass"];
?>
```

$_REQUEST

- The $_REQUEST is a super global variable which is work with both get and post methods.

First.html

```
<html>
<head>
<title>My First HTML web page</title>
</head>
<body>
```

```
        <FORM name="form1" action="result.php" method="GET">
            Name : <input type="text" name="name"><br>
            Password : <input type="password" name="pass"><br>
        </FORM>
    </body>
    </html>
```

result.php

```
    <?php
        echo "Your Username = ". $_REQUEST["name"]. "<br/>";
        echo " Password = ". $_REQUEST["pass"];
    ?>
```

SESSION

- A session is a way to store information (in variables) to be used across multiple pages.
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.

<u>Start a PHP Session</u>

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

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

Get PHP Session Variable Values

- Session variables values get with the PHP global variable: $_SESSION.

```
<?php
session_start();
?>
<!DOCTYPE html>
<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"] . ".";
?>

</body>
</html>
```

How does it work? How does it know it's me?

Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Session state is usually stored in a temporary file. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

To change a session variable, just overwrite it.

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

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

```php
<?php
session_start();
?>
<!DOCTYPE html>
<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."
?>

</body>
</html>
```

COOKIES

- A cookie in PHP is a small file with a maximum size of 4KB that the web server stores on the client computer.
- They are typically used to keep track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time.
- A cookie can only be read from the domain that it has been issued from.
- Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.

Create Cookies with PHP

- A cookie is created with the setcookie () function.

Syntax

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

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

- Name: It is used to set the name of the cookie.
- Value: It is used to set the value of the cookie.
- Expire: It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
- Path: It is used to specify the path on the server for which the cookie will be available.
- Domain: It is used to specify the domain for which the cookie is available.
- Security: It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

- Httponly: it used to indicate that the cookies should be work in http only or not, the value of this parameter is either true or false. Default value is false.

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);
?>
<html>
<body>
    <?php
    if (isset($_COOKIE["Auction_Item"]))
    {
        echo "Auction Item is a  " . $_COOKIE["Auction_Item"];
    }
    else
    {
        echo "No items for auction.";
    }
    ?>
    <p>
        <strong>Note:</strong>
        You might have to reload the page
        to see the value of the cookie.
    </p>

</body>
</html>
```

- It is always advisable to check whether a cookie is set or not before accessing its value. Therefore to check whether a cookie is set or not, the PHP isset() function is used. To check whether a cookie "Auction_Item" is set or not

<u>Accessing Cookie Values</u>

- For accessing a cookie value, the PHP $_COOKIE super global variable is used.
- It is an associative array that contains a record of all the cookies values sent by the browser in the current request.
- The records are stored as a list where the cookie name is used as the key.

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);
?>
<html>
<body>
<?php
    echo "Auction Item is a  " . $_COOKIE["Auction_Item"];
?>
    <p>
        <strong>Note:</strong>
        You might have to reload the page
        to see the value of the cookie.
    </p>

</body>
</html>
```

Deleting Cookies:

- The setcookie() function can be used to delete a cookie.
- If you set the expiration date in past, cookie will be deleted.

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);
?>
<html>
<body>
    <?php
       setcookie("Auction_Item", "", time() - 60);
    ?>
    <?php
        echo "cookie is deleted"
    ?>
    <p>
        <strong>Note:</strong>
        You might have to reload the page
        to see the value of the cookie.
    </p>

</body>
</html>
```

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

Accessing Relational Databases using PHP

PHP and Database

- ***Mysql*** – popular open-source database management system
- ***PHP*** usually works with ***Mysql*** for web-based database applications
- ***XAMPP*** applications—Web-based applications that use ***Lynux/Windows/Mac***, ***Apache***, *Mysql*, and ***php/pear***

What is MySQL?

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

PHP Data Objects (PDO)

What is PDO

- Common interface to any number of database systems.
- Written in C, so you know it's FAST!
- Designed to make use of all the PHP 5.1 features to simplify interface.

Introduction to PDO in PHP

PDO stands for PHP Data Objects, which is a PHP extension providing a consistent method of interacting with databases. It's a database access layer providing a uniform method of access to multiple databases. Here's a short explanation of PDO and its benefits:

PDO (PHP Data Objects): PDO is a database access layer providing a uniform method of access to multiple databases. It provides a data-access abstraction layer, which means that, regardless of which database you're using (MySQL, PostgreSQL, SQLite, etc.), you can interact with the database using the same set of functions.

Benefits of Using PDO:

1. **Database Portability:** PDO allows you to switch between database systems without

changing your code. This means your application can support various databases without major modifications.

2. **Security:** PDO provides prepared statements and parameterized queries, which help prevent SQL injection attacks. Parameters are automatically sanitized, reducing the risk of malicious attacks on your database.
3. **Error Handling:** PDO provides a consistent way to handle errors. It simplifies the process of error detection and handling, making it easier to debug database-related issues.
4. **Performance:** PDO can be faster than some other database access methods in PHP because it can use prepared statements and take advantage of native prepared statement support in certain databases, which can improve performance significantly.
5. **Maintainability:** PDO code tends to be more readable and maintainable than direct SQL queries embedded in the code. It separates the database logic from the application logic, making the codebase cleaner and easier to understand.
6. **Object-Oriented Interface:** PDO provides an object-oriented interface, allowing developers to work with databases using objects and methods, which can lead to more organized and intuitive code.

Databases supported by PDO

1 MySQL

2 PostgreSQL

4 Oracle

5 Firebird

6 MS SQL Server

7 Sybase

8 Informix

9 IBM

10 FreeTDS

11 SQLite

12 Cubrid

13 4D

Example

```
<?php
```

```
$dbHost="localhost";
$dbName="myDB";
$dbUser="root";      //by default root is user name.
$dbPassword="";      //password is blank by default
try{
    $dbConn= new PDO("mysql:host=$dbHost;dbname=$dbName",$dbUser,$dbPassword);
    Echo "Successfully connected with myDB database";
} catch(Exception $e){
echo "Connection failed" . $e->getMessage();
}
?>
```

PDO Functions:

(1) new PDO(): Creates a new PDO instance representing a connection to a database.
Example:

```
$dsn = 'mysql:host=localhost;dbname=mydatabase';
$username = 'username';
$password = 'password';
$pdo = new PDO($dsn, $username, $password);
```

(2) prepare(): Prepares a SQL statement for execution and returns a statement object.

Example:

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username");
```

(3) bindParam() / bindValue(): Binds a parameter to the specified variable name.

Example:

```
$username = 'admin';
$stmt->bindParam(':username', $username);
// OR
// $stmt->bindValue(':username', 'admin');
```

(4) execute(): Executes a prepared statement.

Example:

```
$stmt->execute();
```

(5) fetch(): Fetches a row from a result set.

Example:

```
$row = $stmt->fetch(PDO::FETCH_ASSOC);
echo $row['username'];
```

(6) fetchAll(): Fetches all rows from a result set.

Example:

```
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
foreach ($rows as $row) {
   echo $row['username'];
}
```

(7) rowCount(): Returns the number of rows affected by the last SQL statement.

Example:

```
$rowCount = $stmt->rowCount();
```

(8) lastInsertId(): Returns the ID of the last inserted row or sequence value.

Example:

```
$id = $pdo->lastInsertId();
```

Example

```
try {
   $dsn = 'mysql:host=localhost;dbname=mydatabase';
   $username = 'username';
```

```
    $password = 'password';
    $pdo = new PDO($dsn, $username, $password);

    $stmt = $pdo->prepare("INSERT INTO users (username, email) VALUES (:username, :email)");
    $username = 'admin';
    $email = 'admin@example.com';
    $stmt->bindParam(':username', $username);
    $stmt->bindParam(':email', $email);
    $stmt->execute();

    $id = $pdo->lastInsertId();
    echo "New user inserted with ID: $id";
} catch (PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
```

Error Handling in PDO:

DO provides three error handling modes:

(1) PDO::ERRMODE_SILENT: This mode sets PDO to silently return error codes. You can check for errors by calling **PDO::errorCode()** and **PDO::errorInfo()**.

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);
```

(2) PDO::ERRMODE_WARNING: This mode makes PDO throw a PHP warning for errors. It's useful during development but should be avoided in production.

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
```

(3) PDO::ERRMODE_EXCEPTION: This mode throws exceptions when errors occur, allowing you to catch and handle exceptions in your code.

$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

Example of Error Handling with PDO:

```
try {
   // Database connection
   $pdo = new PDO('mysql:host=localhost;dbname=mydatabase', 'username', 'password');

   // Set the PDO error mode to exception
   $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

   // SQL query
   $sql = "SELECT * FROM users WHERE id = :id";
   $stmt = $pdo->prepare($sql);

   // Bind parameters and execute the query
   $id = 1;
   $stmt->bindParam(':id', $id, PDO::PARAM_INT);
   $stmt->execute();

   // Fetch results
   $user = $stmt->fetch(PDO::FETCH_ASSOC);

   // Handle the fetched data (for example, print it)
   print_r($user);
} catch (PDOException $e) {
   // Handle PDO exceptions
   echo "Error: " . $e->getMessage();
} catch (Exception $e) {
   // Handle other types of exceptions
   echo "Error: " . $e->getMessage();
}
```

Unit-VI PHP Utilities

Introduction to the file input element

- The <input> element with the type="file" allows you to select one or more files from their storage and upload them to the server via the form submission.
- The following shows the file input element:

  <input type="file" id="file" name="file">
- The value of the <input> element will hold the path to the selected file. To upload multiple files, you add the multiple attribute to the <input> element like this:

  <input type="file" id="file" name="file" multiple>
- In this case, the value attribute will hold the path of the first file in the selected file list.
- To allow certain file types to be uploaded, you use the accept attribute. The value of the accept attribute is a unique file type specifier, which can be:
  - A valid case-insensitive file name extension e.g., .jpg, .pdf, .txt
  - A valid MIME type string or a string like image/* (any image file), video/* (any video file), audio/* (any audio file).
- If you use multiple file type specifiers, you need to separate them using a comma (,). For example, the following setting allows you to upload only .png and .jpeg images:

  <input type="file" accept="image/png, image/jpeg" name="file">
- The <form> element that contains the file input element must have the enctype attribute with the value multipart/form-data:

  <form enctype="multipart/form-data" action="index.php" method="post">
  </form>

File Uploading

- PHP allows you to upload single and multiple files.
- 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 super global variable $_FILES used for file uploading.

$_FILES super global variable

- 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 input type file have fileupload value to the name attribute in html.

#	$_FILES Parameters	Description
1	$_FILES['fileupload']['name']	The original name of the file on the client machine.
2	$_FILES['fileupload']['type']	The mime type of the file, if the browser provided this information. An example would be "image/gif". This mime type is however not checked on the PHP side and therefore don't take its value for granted.
3	$_FILES['fileupload']['size']	The size, in bytes, of the uploaded file.
4	$_FILES['fileupload']['tmp_name']	The temporary filename of the file in which the uploaded file was stored on the server.
5	$_FILES['fileupload']['error']	The error code associated with this file upload.

move_uploaded_file() function

- When a file is uploaded successfully, it is stored in a temporary directory on the server.
- move_uploaded_file() function to move the file from the temporary directory to another one.
- The move_uploaded_file() function accepts two arguments:
  - filename: is the file name of the uploaded file which is $_FILES['file']['tmp_name'].
  - destination: is the destination of the moved file.
- The move_uploaded_file() function returns true if it moves the file successfully; otherwise, it returns false.
- Syntax: **bool move_uploaded_file (string $filename , string $destination)**

Pathinfo() function

- Pathinfo() returns information about a file path
- Syntax: ***pathinfo(string $path, int $flags = PATHINFO_ALL): array|string***
- pathinfo() returns information about path: either an associative array or a string, depending on flags.
- Parameters:
  - Path: The path to be parsed.

- Flags: If present, specifies a specific element to be returned; one of PATHINFO_DIRNAME, PATHINFO_BASENAME, PATHINFO_EXTENSION or PATHINFO_FILENAME. If flags are not specified, returns all available elements.

Important settings for file uploads in the php.ini file:

- file_uploads
  - The file_upload directive should be On to allow file upload. It defaults to On.
- upload_max_filesize
  - The upload_max_filesize specifies the maximum size of the uploaded file. By default, it's 2M (MB). If you get an error saying that the file exceeds upload_max_filesize, you need to increase this value.
- upload_tmp_dir
  - The upload_tmp_dir specifies the directory that stores the uploaded files temporarily.
- post_max_size
  - The post_max_size specifies the maximum size of the POST data. Because you'll upload files with the POST request, you need to make sure that the post_max_size is greater than upload_max_size.
- max_file_uploads
  - The max_file_uploads directive limits the number of files that you can upload at a time.

Getting information from web server

- $_SERVER super global variable is used to get the information of server and execution environment.

#	Elements/Code	Description
1	$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
2	$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
3	$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
4	$_SERVER['SERVER_NAME']	Returns the name of the host server .
5	$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)

6	$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
7	$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
8	$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
9	$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
10	$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
11	$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
12	$_SERVER['HTTP_HOST']	Returns the Host header from the current request
13	$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
14	$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
15	$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
16	$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
17	$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
18	$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
19	$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script

		runs on a virtual host, it will be the value defined for that virtual host)
20	$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
21	$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
22	$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
23	$_SERVER['SCRIPT_NAME']	Returns the path of the current script
24	$_SERVER['SCRIPT_URI']	Returns the URI of the current page

What is AJAX?

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.
- Conventional web application transmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

The XMLHttpRequest Object

- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

- All modern browsers (Chrome, Firefox, Edge (and IE7+), Safari, Opera) have a built-in XMLHttpRequest object.
- Syntax for creating an XMLHttpRequest object:
    - variable = new XMLHttpRequest();
- Example

- var xhttp = new XMLHttpRequest();

AJAX - Send a Request to a Server

- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

#	Method	Description
1	open(method, url, async)	specifies the type of request. **method:** the type of request: GET or POST **url:** the server (file) location **async:** true (asynchronous) or false (synchronous)
2	send()	Sends the request to the server

AJAX - Server Response

- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes.
- The status property and the statusText property holds the status of the XMLHttpRequest object.

#	Properties	Description
1	onreadystatechange	Defines a function to be called when the readyState property changes
2	readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
3	Status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
4	statusText	Returns the status-text (e.g. "OK" or "Not Found")

- The onreadystatechange function is called every time the readyState changes.
- When readyState is 4 and status is 200, the response is ready:

Example: Test.php

```
<html>
  <head>
    <title>PHP MVC Frameworks - Search Engine</title>
     <script type="text/javascript">
       function showName(str)
       {
          if (str.length == 0){ //exit function if nothing has been typed in the textbox
             document.getElementById("txtName").innerHTML=""; //clear previous results
             return;
          }

          if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp=new XMLHttpRequest();
          } else {// code for IE6, IE5
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

          }
          xmlhttp.onreadystatechange=function() {
             if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
                document.getElementById("txtName").innerHTML=xmlhttp.responseText;
             }
          }

       xmlhttp.open("GET","frameworks.php?name="+str,true);
       xmlhttp.send();

    }
    </script>
  </head>
  <body>
    <h2>PHP MVC Frameworks - Search Engine</h2>
    <p><b>Type the first letter of the PHP MVC Framework</b></p>
    <form method="POST" action="">
```

```
        <p><input type="text" size="40" id="txtHint"  onkeyup="showName(this.value)"></p>
      </form>
      <p>Matches: <span id="txtName"></span></p>
    </body>
  </html>
```

Framework.php

```
<?php
 $frameworks = array ("CodeIgniter","Zend Framework","Cake PHP", "Kohana", "Laravel", "Symfony",
"Laminas Project") ;
  $name = $_GET["name"];
  if (strlen($name) > 0) {
    $match = "";
    for ($i = 0; $i < count($frameworks); $i++) {
      if (strtolower($name) == strtolower(substr($frameworks[$i], 0, strlen($name)))) {
        if ($match == "") {
          $match = $frameworks[$i];
        } else {
          $match = $match . " , " . $frameworks[$i];
        }        }
    }   }
  echo ($match == "") ? 'no match found' : $match;
?>
```