MODULE-01

Introduction to PHP: Introduction to PHP, History and Features of PHP, Installation & Configuration of PHP, Embedding PHP code in Your Web Pages, Understanding PHP,HTML and White Space, Writing Comments in PHP, Sending Data to the Web Browser,Data types in PHP, Keywords in PHP, Using Variables, Constants in PHP, Expressions in PHP, Operators in PHP.

INTRODUCTION TO PHP: Introduction to PHP

- PHP stands for Hypertext Pre-processor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.
- PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.
- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best langua ge to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. **For example** - Registrat ion form.
- PHP is a recursive acronym for "PHP: Hypertext Pre-processor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e- commerce sites.

- It is integrated with a number of popular databases, including MySQL, Postgre SQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The My SQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first t.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.
- In PHP Soo many programmers are choosing the WEB DEVELOPMENT

WEB DEVELOPMENT:

PHP is widely used in web development nowadays. PHP can develop dynamic websites easily. However, you must have the basic the knowledge of following technologies for web development as well.

1. HTML
2. CSS
3. JavaScript
4. PHP
5. Ajax
6. XML and JSON
7. JQuery

1) **HTML** – HTML is used to design static webpage.
2) **CSS -** CSS helps to make the webpage content more effective and attractive.
3) **JavaScript -** JavaScript is used to design an interactive website.
4) **PHP**-Server side scripting language.
5) **AJAX- AJAX** allows web pages to be updated asynchronously by exchanging data with a web server.
6) **XML AND JSON-** JSON and XML can be used as data interchange formats by many different programming languages.
7) **JQUERY- jQuery** is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animations, and Ajax.

PHP AND MYSQL
HISTROY PHP

- In the year of 1993 Dennis Canadians is invented by " personnel Home page tools" In (PHP).
- In later 1994 onwards Personnel Home page Tools are Upgraded by Hypertext pre-processor

Rasmus Lerdorf

Rasmus Lerdorf (born 22 November 1968) is a programmer. He co-authored and inspired the PHP scripting language, authoring the first two versions of the language and participat ing in the development of later versions led by a group of developers including Jim Winstead (who later created blo.gs), Stig Bakken, Shane Caraveo, Andi Gutmans, and Zeev Suraski. He continues to contribute to the project.

- PHP started out as a small open-source project that evolved gradually as more and more people found out how useful it was.
- Rasmus Lerdorf released the first version of PHP way back in 1994. At that time, PHP stood for Personal Home Page, as he used it to maintain his personal homepage.
- Later on, he added database support and called it as "Personal Home Page/Forms Interpreter" or PHP/FI, which could be used to build simple, dynamic web applicatio ns.
- PHP gained popularity, Lerdorf continued to add functionality to it, such as form handling and database interaction. In 1995, he released the source code for PHP/FI (Personal Home Page/Forms Interpreter) for the public to use. This marked the beginning of PHP's journey as a widely-used server-side scripting language for web development.

VERSIONS OF PHP:

- **PHP 1.0:**It First version was released version **1.0.**
- **PHP 2.0:**In April 1996, Rasmus introduced PHP/FI. Included built-in support for DBM, MySQL, and Postgres95 databases, cookies,fre user-defined function support. PHP/FI was given the **version 2.0** status.
- **PHP3.0:** Hypertext Pre-processor – **PHP 3.0** version came about when Zeev Suraski and Andi Gutmans rewrote the PHP parser and acquired the present-day acronym. It provided a mature interface for multiple databases, protocols and APIs, object-oriented programming support, and consistent language syntax.
- **PHP 4.0** was released in May 2000 powered by Zend Engine. It had support for many web servers, HTTP sessions, output buffering, secure ways of handling user input and several new language constructs.
- **PHP 5.0** was released in July 2004. It is mainly driven by its core, the Zend Engine 2.0 with a new object model and dozens of other new features. PHP's developme nt team includes dozens of developers and others working on PHP-related and supporting projects such as PEAR, PECL, and documentation.
- **PHP 7.0** was released in Dec 2015. This was originally dubbed PHP next generation (phpng). Developers reworked Zend Engine is called Zend Engine 3. Some of the important features of PHP 7 include its improved performance, reduced memory usage, Return and Scalar Type Declarations and Anonymous Classes.
- **PHP 8.0** was released on 26 November 2020. This is a major version having many significant improvements from its previous versions. One standout feature is Just-in-time compilation (JIT) that can provide substantial performance improvements. The latest version of PHP is 8.2.8, released on July 4th, 2023.
- **PHP 8.2.3:**It is the latest version of php released in the year of 18th Jan 2024.

INSTALLATION &CONFIGURATION OF PHP

How do I install XAMPP?

- Although an installer is available from php.net, I would recommend the manual installation if you already have a web server configured and running.
- Manual Installation
- Manual installation offers several benefits:
- backing up, reinstalling, or moving the web server can be achieved in seconds (see 8 Tips for Surviving PC Failure) and have more control over PHP and Apache configuration.

Step1: Download the files

Step2: Extract the files

- We will install the PHP files to C:\php, so create that folder and extract the contents of the ZIP file into it.
- PHP can be installed anywhere on your system, but you will need to change the paths referenced in the following steps.

XAMPP for Windows exists in three different flavors:

Installer:

Probably the easiest way to install XAMPP.

ZIP:

For purists: XAMPP as ordinary ZIP archive.

7zip:

For purists with low bandwidth: XAMPP as 7zip archive.

Note: If you extract the files, there can be false-positives virus warnings.

Using the installer:

The XAMPP control panel for start/stop Apache, MySQL, FileZilla & Mercury or install these server as services.

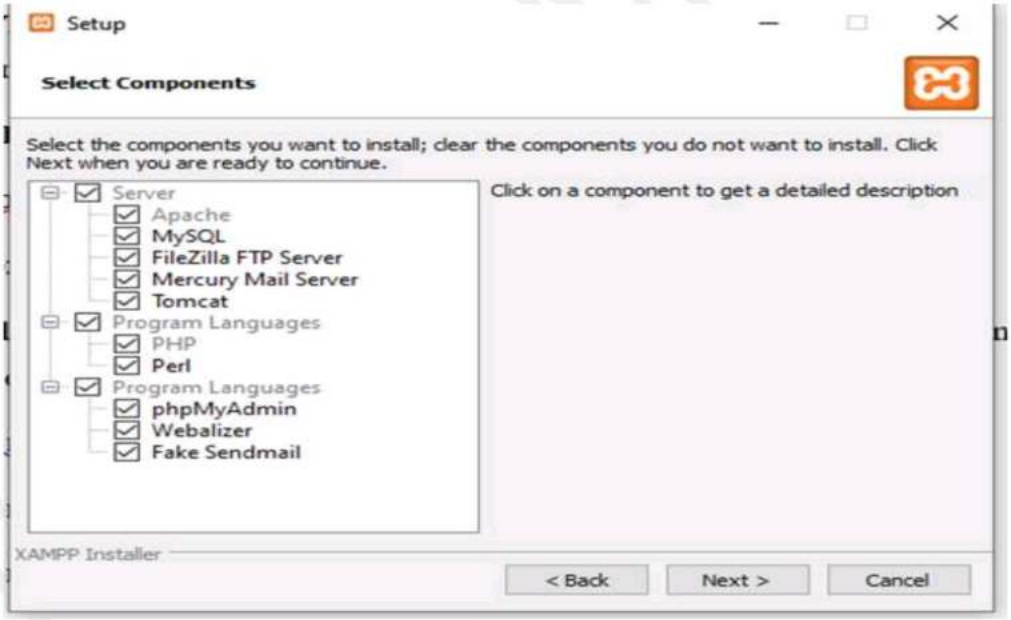
Installing from ZIP

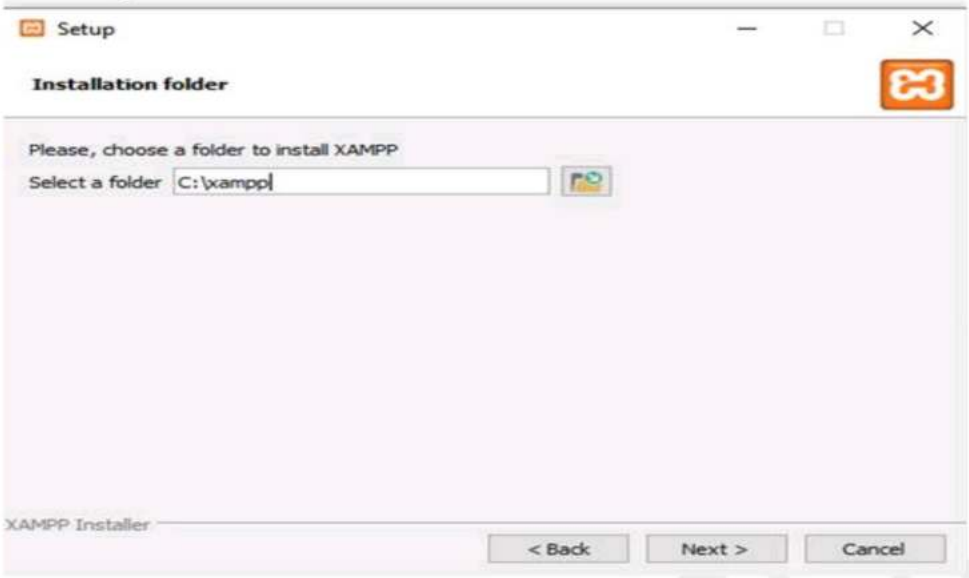
Unzip the zip archives into the folder of your choice. XAMPP is extracting to the subdirectory "C:\xampp" below the selected target directory. Now start the file "setup_xampp.bat", to adjust the XAMPP configuration to your system.

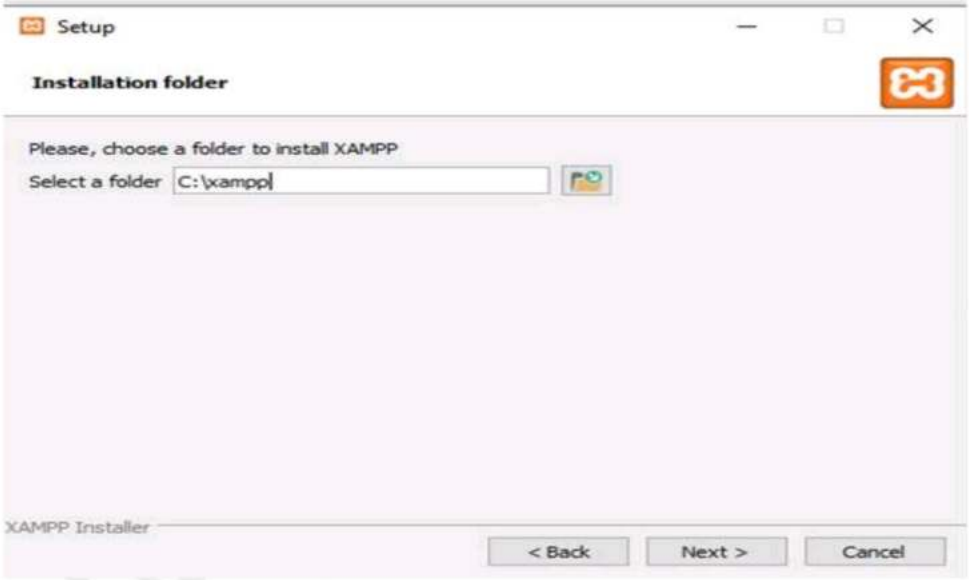
If you choose a root directory "C:\" as target, you must not start "setup_xampp.bat".

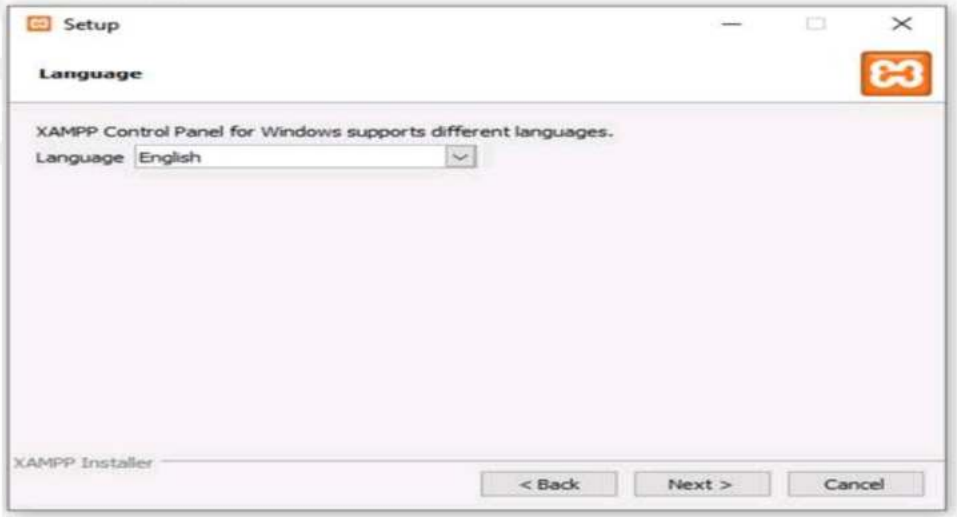
PHP AND MYSQL

Like with the installer version, you can now use the "XAMPP Control Panel" for additiona l tasks.




Setup
Installation folder
Please, choose a folder to install XAMPP
Select a folder C:\xampp\
XAMPP Installer
< Back
Next >
Cancel


Setup
Installation folder
Please, choose a folder to install XAMPP
Select a folder C:\xampp\
XAMPP Installer
< Back
Next >
Cancel


Setup
Language
XAMPP Control Panel for Windows supports different languages.
Language English
XAMPP Installer
< Back
Next >
Cancel

HOW DO I START AND STOP XAMPP?

The universal control center is the "XAMPP Control Panel" (thanks www.nat32.com). It is started with:

\xampp\xampp-control.exe

You can also use some batchfiles to start/stop the servers:

- Apache & MySQL start: \xampp\xampp_start.exe
- Apache & MySQL stop: \xampp\xampp_stop.exe
- Apache start: \xampp\apache_start.bat
- Apache stop: \xampp\apache_stop.bat
- MySQL start: \xampp\mysql_start.bat
- MySQL stop: \xampp\mysql_stop.bat
- Mercury Mailserver start: \xampp\mercury_start.bat
- Mercury Mailserver stop: \xampp\mercury_stop.bat
- FileZilla Server start: \xampp\filezilla_start.bat
- FileZilla Server stop: \xampp\filezilla_stop.bat

How can I test that everything worked?

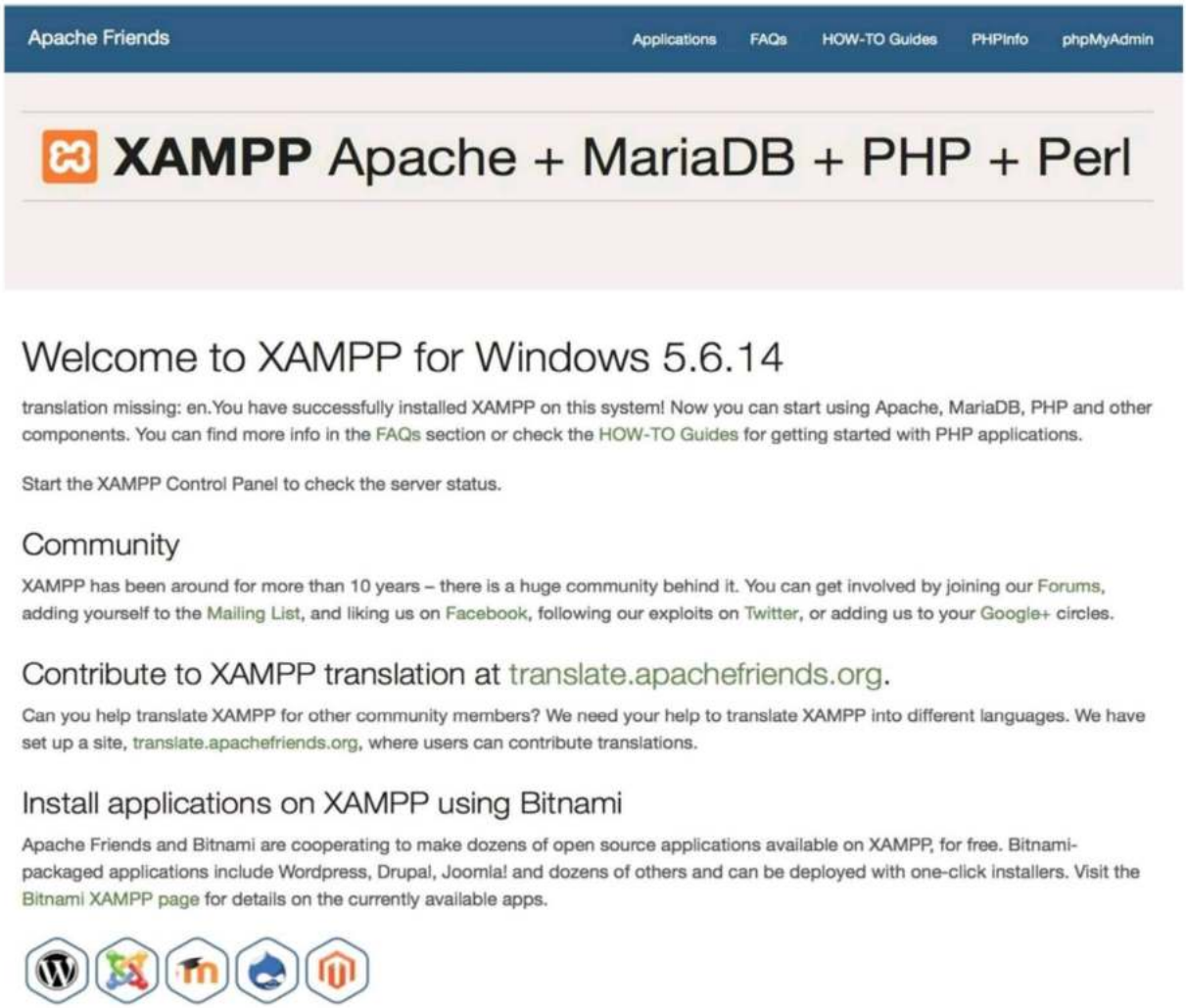
Type in the following URL at your favourite web browser:

http://localhost/

or

http://127.0.0.1/

You should see the XAMPP start page, as shown below.



Where should I place my web content?

The main directory for all WWW documents is \xampp\htdocs. If you put a file "test.html" in this directory, you can access it with the URI "http://localhost/test.html".

And "test.php"? Just use "http://localhost/test.php". A simple test script can be:

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

A new subdirectory for your web? Just make a new directory (e.g. "new") inside the directory "\xampp\htdocs" (best without whitespaces and only ASCII), create a test file in this directory and access it with http://localhost/new/test.php

PHP:
Executable: \xampp\htdocs and \xampp\cgi-bin
Allowed endings: .php
=> basic package

Perl:
Executable: \xampp\htdocs and \xampp\cgi-bin
Allowed endings: .pl
=> basic package

Why can't XAMPP work on Windows XP SP2?

Microsoft delivers a better firewall with service pack 2 (SP2), which starts automatically. This firewall now blocks the necessary ports 80 (http) and 443 (https) and Apache can't start.

The fast solution:

Disable the Microsoft firewall with the toolbar and try to start XAMPP onces more. The better solution is to define an exception within the security center.

The following ports are used for basic functionality:

Apache (HTTP): Port 80
Apache (WebDAV): Port 81
Apache (HTTPS): Port 443
MySQL: Port 3306
FileZilla (FTP): Port 21
FileZilla (Admin): Port 14147
Mercury (SMTP): Port 25
Mercury (POP3): Port 110
Mercury (IMAP): Port 143
Mercury (HTTP): Port 2224
Mercury (Finger): Port 79
Mercury (PH): Port 105
Mercury (PopPass): Port 106
Tomcat (AJP/1.3): Port 8009
Tomcat (HTTP): Port 8080

FEATURES OF PHP

1.Open Source: PHP is open source, meaning it's freely available for anyone to use and modify. This has contributed to its widespread adoption and the vast ecosystem of PHP libraries and frameworks available for web development.

2. **Embedded in HTML**: PHP code can be embedded directly into HTML, allowing for the seamless integration of dynamic content within web pages. PHP code is typically enclosed within **<?php ... ?>** tags.

3. **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.

4. **Freely/Simply typed language:** PHP is primarily considered a loosely typed langua ge, meaning that variable types are not explicitly declared and can change dynamically based on the value assigned to them. This can lead to flexible but sometimes unpredictable behaviour.

5. **Security:** PHP provides various features for enhancing the security of web applicatio ns, including input validation functions, secure database connection methods, and measures to prevent common vulnerabilities like SQL injection and cross-site scripting (XSS).

6. **Web Server Support/Dynamic Webpage:** PHP is widely supported by various web servers, making it a versatile choice for dynamic web development. Some of the most commonly used web servers that support PHP**.** Apache HTTP Server, LiteSpeed Web Server.

7. **Database support:** PHP has robust support for interacting with various databases, includ ing MySQL, PostgreSQL, SQLite, Oracle, and others. This enables developers to build database-driven applications for storing, retrieving, and manipulating data.

8. **Platform Independent**: PHP is indeed platform-independent, as it runs on various operating systems, including Windows, macOS, Linux, and Unix. This means that PHP applicatio ns developed on one operating system can be easily deployed and executed on other supported operating systems without significant modifications.

9. **Familiarity with Syntax**: PHP has a syntax that is easy to understand and learn, especially for those familiar with HTML and similar languages. Its syntax is relatively straightforward, making it accessible to programmers of various skill levels.

10. **Interpreted Language**: PHP is an interpreted language, meaning that the source code is executed line by line by an interpreter at runtime, rather than being compiled into machine code beforehand. This enables rapid development and testing cycles since changes to the code can be immediately reflected without the need for compilation.

11. **Object-Oriented Language** : PHP is both a procedural and object-oriented language. While it initially started as a procedural language, support for object-oriented programming (OOP) was introduced in PHP 4 and significantly improved in later versions. This allows developers to organize their code into classes and objects, promoting modularity, reusability, and maintainability in their PHP applications.

PHP AND MYSQL

EMBEDDING PHP CODE IN YOUR WEB PAGES:

Embedding PHP code in your web pages allows you to create dynamic content and interact with databases, handle forms, and perform other server-side tasks. PHP code is typically embedded within HTML using special delimiters **<?php** and **?>**.

- Embedding PHP code in web pages allows for the creation of dynamic web content by mixing PHP scripting with HTML markup.

Uses Embedding Php Code In Your Web Pages:

Embedding PHP code in your web pages offers numerous benefits and use cases, allowing you to create dynamic, interactive, and data-driven websites.

1. **Dynamic Content Generation:** PHP enables you to generate dynamic content based on user input, database queries, or other conditions. For example, you can personalize greetings, display the current date and time, or fetch and display data from a database.
2. **Form Handling:** PHP can handle form submissions by processing form data, validating inputs, and performing actions based on user input. This includes tasks such as user authentication, form validation, and data storage.
3. **Database Interaction:** PHP can interact with databases such as MySQL, PostgreSQL, or SQLite, allowing you to retrieve, insert, update, and delete data from databases. You can dynamically generate content based on database queries or store user-submit ted data in a database.
4. **Session Management:** PHP enables you to manage user sessions, track user interactions, and maintain user-specific data across multiple page requests. This is useful for implementing features like user authentication, user preferences, and shopping carts.
5. **Conditional Content Display:** PHP allows you to display different content based on conditions such as user permissions, user preferences, or environmental factors. This includes showing or hiding content, altering content based on user roles, or displaying messages based on certain conditions
6. **Template Integration:** PHP can be used to integrate dynamic content with HTML templates, allowing for consistent design and layout across multiple pages. You can separate the presentation layer (HTML) from the logic layer (PHP), making it easier to maintain and update your website.
7. **Error Handling:** PHP provides mechanisms for error handling and debugging, allowing you to catch and handle errors gracefully. You can display custom error

messages, log errors for debugging purposes, or redirect users to error pages when errors occur.

8. **Content Management Systems (CMS):** PHP powers many popular CMS platforms like WordPress, Joomla, and Drupal. These systems use PHP to dynamically generate and manage website content, including articles, pages, comments, and media.

Example: Source code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Embedded PHP Example</title>
</head>
<body>
    <h1>Embedded PHP Example</h1>

    <p>Current date and time: <?php echo date("Y-m-d H:i:s"); ?></p>

    <?php
        // PHP code block
        $name = "SMITHA";
        echo "<p>Hello, $name!</p>";
    ?>

    <?php if ($name === "Smitha"): ?>
        <p>Welcome back, smitha!</p>
    <?php else: ?>
        <p>Hello, guest!</p>
    <?php endif; ?>

</body>
</html>
```

OUTPUT:

Embedded PHP Example

Current date and time: 2024-04-02 18:44:13

Hello, SMITHA!

Hello, guest!

In the above example:

PHP code is enclosed within **<?php ... ?>** tags.

The **date()** function retrieves the current date and time and outputs it within the HTML.

A variable **$name** is declared and then echoed within an HTML paragraph.

A conditional statement is used to display different messages based on the value of the **$name** variable.

UNDERSTANDING PHP:

Characteristics of php

PHP (Hypertext Pre-processor) is a popular server-side scripting language known for its versatility and ease of use in web development. Some key characteristics of PHP include:

1. **Server-side scripting**: PHP code is executed on the server, generating HTML which is then sent to the client's browser. This allows for dynamic content generation and interaction with databases.
2. **Cross-platform compatibility**: PHP runs on various platforms, including Windows, Linux, macOS, and Unix, making it accessible for developers using different operating systems.
3. **Open source** : PHP is free to use and has a large community of developers contribut ing to its development and maintenance. This results in a wealth of resources, libraries, and frameworks available for PHP development.
4. **Ease of learning**: PHP syntax is similar to other C-style languages like C, Java, and JavaScript, making it relatively easy for developers to learn and use, especially for those already familiar with these languages.
5. **Integration with databases**: PHP has built-in support for interacting with various databases, including MySQL, PostgreSQL, SQLite, and others, allowing developers to create dynamic, database-driven websites and applications.
6. **Extensive library support**: PHP has a vast ecosystem of libraries and extensions that provide additional functionality for tasks such as image processing, PDF generatio n, encryption, and more.
7. **Scalability**: PHP applications can scale effectively to handle increased traffic and workload by leveraging techniques like caching, load balancing, and optimizing code.
8. **Community support**: PHP has a large and active community of developers who provide support, share knowledge, and contribute to the improvement of the langua ge and its ecosystem through forums, online communities, and conferences.

Applications of PHP

PHP is widely used in various applications across web development. Some common applications include:

1. **Web Development**: PHP is primarily used for creating dynamic and interactive websites. It can handle tasks such as user authentication, form processing, database interactions, session management, and content management systems (CMS) development.
2. **E-commerce** : Many e-commerce platforms, including WooCommerce (WordPress plugin), Magento, and OpenCart, are built using PHP. PHP enables developers to create robust and scalable online stores with features like product catalog manageme nt, shopping cart functionality, payment gateway integration, and order processing.
3. **Content Management Systems (CMS)**: PHP powers popular CMS platforms like WordPress, Drupal, and Joomla. These platforms allow users to create, manage, and publish digital content, such as articles, blogs, and multimedia, without needing extensive technical knowledge.
4. **Social Media Platforms**: PHP is used in developing social networking sites and platforms like Facebook, where it handles various functionalities such as user profiles, news feeds, messaging systems, and notifications.
5. **Web Applications**: PHP is utilized in building a wide range of web applicatio ns, including project management tools, customer relationship management (CRM) systems, online forums, wikis, and file management systems.
6. **API Development**: PHP can be used to develop backend APIs (Applicat ion Programming Interfaces) for mobile applications, allowing them to communicate with server-side resources and perform tasks like user authentication, data retrieval, and updates.
7. **Real-time Chat Applications**: PHP can be combined with technologies like WebSocket or AJAX to create real-time chat applications, enabling users to communicate instantly with each other through text, voice, or video messages.
8. **Data Processing and Manipulation**: PHP is often employed for data processing tasks, such as parsing and manipulating XML or JSON data, generating reports, handling file uploads, and interacting with external APIs and services.
9. **Web Services** : PHP can be used to develop RESTful APIs and SOAP web services, enabling communication between different systems and allowing them to exchange data and perform actions over the web.

ADVANTAGES AND DISADVANTAGES OF PHP

Advantages:

1. **Ease of Use** : PHP is relatively easy to learn and use, especially for beginners. Its syntax is similar to C and Perl, making it accessible to developers with programmin g experience in other languages.
2. **Open Source**: PHP is an open-source language, which means it's free to use, distribute, and modify. This contributes to its widespread adoption and a large community of developers who contribute to its development and provide support.
3. **Platform Independence** : PHP runs on various operating systems, including Windows, Linux, macOS, and Unix. This ensures compatibility across different platforms, making it versatile for web development.
4. **Integration with Databases**: PHP has built-in support for interacting with databases, including MySQL, PostgreSQL, SQLite, and others. This makes it suitable for building database-driven web applications and dynamic websites.
5. **Large Ecosystem**: PHP has a vast ecosystem of libraries, frameworks, and extensions that extend its functionality and simplify development tasks. Popular frameworks like Laravel, Symfony, and Code Igniter provide pre-built components and conventions for building web applications efficiently
6. **Fast Execution**: PHP is a server-side scripting language, which means the code is executed on the server before being sent to the client's browser. This can result in faster loading times compared to client-side scripting languages like JavaScript

Disadvantages:

1. **Security Concerns** : PHP's open nature and popularity make it a target for security vulnerabilities. Poorly written code, improper configuration, and outdated versions of PHP can lead to security vulnerabilities such as SQL injection, cross-site scripting (XSS), and remote code execution.
2. **Inconsistencies**: PHP's evolution over the years has led to inconsistencies in its function names, parameter orders, and behaviours. This can sometimes make it challenging for developers to maintain code across different PHP versions or frameworks.
3. **Scalability**: While PHP is suitable for building small to medium-sized web applications, it may face scalability challenges when handling large-scale projects with high traffic and complex requirements. Proper architectural design and optimization techniques are necessary to ensure scalability.

4. **Performance** : Compared to some other programming languages like Java or Go, PHP may not offer the same level of performance and efficiency, especially for computationally intensive tasks. However, advancements in PHP runtime engines like Zend Engine and PHP-FPM have improved its performance significantly.
5. **Community Support**: While PHP has a large community of developers and resources available, the quality and reliability of community-contributed libraries and frameworks may vary. It's essential to carefully evaluate third-party components and ensure they meet the project's requirements and security standards.

SYNTAX OF PHP

The syntax of PHP is relatively straightforward, resembling that of C, Java, and Perl. Here's an overview of the basic syntax elements in PHP:

Opening and Closing Tags:

PHP code is typically enclosed within **<?php ?>** tags.

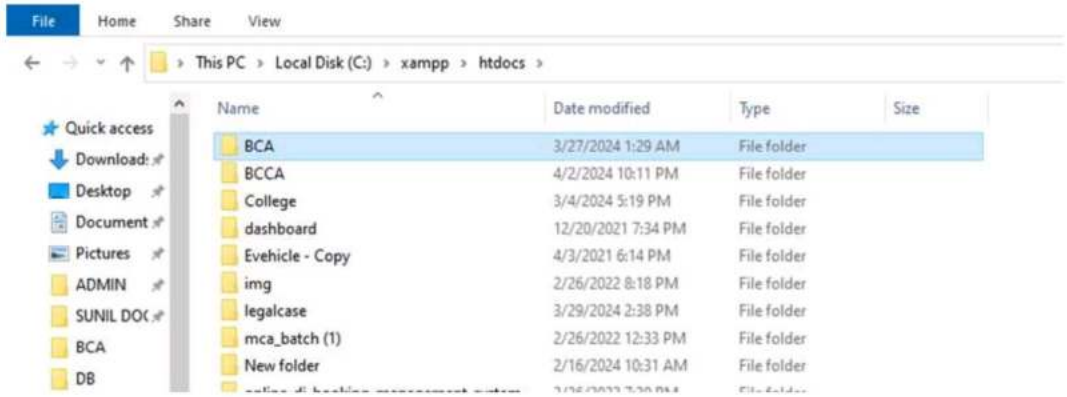
For example:

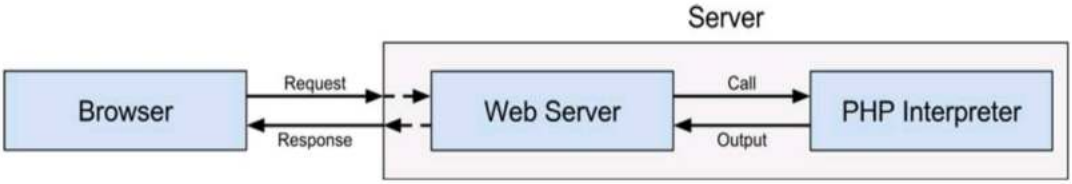
```
<?php

// PHP code goes here

?>
```

Note:

- To take any text editor like note pad ,notepad++,sublimetext,visual studio code to write the code in file .
- To save the file name is filename.php
- The file Extension of php is the .php
- To save the file location in the xampp htdocs folder create one folder and save the php file in htdocs created folder.

  Example: To create one folder BCA in htdocs folder





HOW TO RUN PHP FILE IN XAMPP SOFTWARE PACKAGE

http://localhost/BCA/prg1.php

- Here localhost is the server address alternatively using 127.0.0.0.1/
- Here http is the server or protocol
- Localhost is the server name
- BCA is the folder name
- Prg1.php is the php file

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

OUTPUT: `Hello, World!";`

HTML AND WHITE SPACE:

HTML is used in PHP in various ways to create the structure, layout, and presentation of web pages. Some common uses of HTML in PHP include:

1. **Page Structure** : HTML is used to define the basic structure of web pages, includ ing elements like **<html>**, **<head>**, **<title>**, **<body>**, and others. PHP scripts often generate HTML code dynamically to produce different page structures based on conditions or data.
2. **Forms**: HTML forms are used to collect user input, such as login credentials, registration details, search queries, or feedback. PHP scripts handle form submissions by accessing form data through the **$_POST** or **$_GET** superglobal arrays, and then process and validate the input as needed.
3. **Displaying Data**: PHP scripts often retrieve data from databases, APIs, or other sources, and then use HTML to format and display this data on web pages. This may involve using HTML elements like **<table>**, **<ul>**, **<ol>**, ** **, ****, etc., to organize and present the data in a user-friendly manner.
4. **Dynamic Content**: HTML is used to display dynamic content generated by PHP scripts. This content may include user-specific information, such as profile details, shopping cart items, or personalized recommendations, which are dynamically inserted into HTML templates using PHP variables or loops.

5. **Links and Navigation**: HTML **<a>** tags are used to create hyperlinks that allow users to navigate between different pages or sections of a website. PHP scripts can dynamically generate these links based on various factors, such as user permissio ns, page content, or database records.
6. **Images and Media**: HTML **<img>**, **<audio>**, and **<video>** tags are used to display images, audio files, and videos on web pages. PHP scripts may dynamically generate URLs for these media files based on user preferences or database entries.
7. **HTML Templates** : PHP often uses HTML templates to separate the presentation layer from the business logic. HTML templates contain static HTML code along with placeholders or template tags that are replaced with dynamic content by PHP scripts during runtime.
8. **Client-Side Interactivity**: HTML can include JavaScript code to add client-side interactivity to web pages, such as form validation, animations, or AJAX requests. PHP scripts may generate HTML with embedded JavaScript code to enhance the user experience.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PHP Embedded in HTML</title>
</head>
<body>
    <h1>Welcome to My Website</h1>

    <p>This is a paragraph written in HTML.</p>

    <?php
        // PHP code embedded within HTML
        $name = "Radha";
        $age = 30;

        echo "<p>Hello, $name! You are $age years old.</p>";
    ?>

    <p>Back to HTML again.</p>
</body>
</html>
```

OUTPUT: Welcome to My Website

This is a paragraph written in HTML.

Hello, Radha! You are 30 years old.

Back to HTML again.

In this example:

The HTML tags (**<!DOCTYPE html>**, **<html>**, **<head>**, **<title>**, **<body>**, **<h1>**, **<p>**, etc.) define the structure and content of the web page.

Inside the **<body>** tag, there is a mixture of HTML and PHP code.

The PHP code (**<?php ... ?>**) is used to dynamically generate content. In this case, it defines variables **$name** and **$age**, and then uses them to generate a personalized greeting within a paragraph (**<p>**) tag.

The PHP code is processed on the server before the HTML is sent to the client's browser, so the browser receives only the resulting HTML.

WHITESPACE IN PHP:

In PHP, whitespace refers to characters that are used for spacing and formatting purposes, such as spaces, tabs, and newlines. Whitespace in PHP is typically used to improve code readability and organization.

Here are some common uses of whitespace in PHP:

1. **Indentation**: Whitespace is often used to indent code blocks within control structures (e.g., **if**, **for**, **while**), making the code easier to read and understand.

```
if ($condition) {
    // indented code block
    // more indented code
}
```

2. **Separating Tokens**: Whitespace is used to separate tokens in PHP code, such as variables, operators, function names, etc.

```
$variable = $value + $another_variable;
```

3. **Whitespace between Function Arguments**: Whitespace is used to separate arguments in function calls.

```
echo calculateSum(2, 3);
```

4. **Code Readability**: Whitespace is used to improve the readability of code by adding space between elements. Proper indentation and spacing make code easier to understand and maintain.
5. **Token Separation**: Whitespace is used to separate tokens such as keywords, identifiers, operators, and literals in the code. For example, in PHP, whitespace separates variables from operators and operands.
6. **Indentation**: Indentation, achieved with whitespace, is used to visually represent the structure of the code. It helps identify code blocks, control flow structures, and nested elements.
7. **Blank Lines**: Blank lines, created with whitespace characters, are used to separate different sections of code logically. They improve code organization and readabilit y, especially in larger files or complex scripts.
8. **String Formatting** : Whitespace can be used within strings to control formatting, such as adding spaces between words or aligning text in a particular way.
9. **Comments**: Whitespace is often included within comments to enhance readability. Properly formatted comments with appropriate whitespace make code documentat ion and explanations more understandable.
10. **HTML Output**: In web development with PHP, whitespace affects the formatting and layout of HTML output. Careful use of whitespace ensures well-formatted HTML for better presentation and readability in web pages.

Example program.

```php
<?php
// Define variables
    $fruit     = "Apple";
    $quantity  = 5;
    $price     = 1.25;

// Calculate total cost
$totalCost = $quantity * $price;

// Output the invoice
    echo  "Invoice:\n";
    echo "  \n";
echo "Item:     " . $fruit . "\n";
    echo "Quantity: " . $quantity . "\n";
echo "Price: $" . $price . "\n";
echo "          \n";
    echo "Total:   $" . $totalCost . "\n";

?>
```

OUTPUT:

Invoice: -------- Item:

Apple Quantity: 5

Price: $1.25 --------

Total: $6.25

WRITING COMMENTS IN PHP:

- Comments in programming languages are non-executable text that programmers include within their code to provide explanations, documentation, or annotations.
- Comments are ignored by the interpreter or compiler and serve as a form of communication between developers, helping to improve code readability and maintainability.

USES OF COMMENTS IN PHP

1. **Documentation**: Comments are used to document code, providing explanatio ns about how it works, why certain decisions were made, or any important information that other developers may need to understand when reading the code. This helps improve code readability and maintainability, making it easier for developers to work with and modify the code in the future.
2. **Debugging**: Comments can be used to temporarily disable lines or blocks of code during debugging or testing without deleting them. This allows developers to quickly identify and isolate issues in the code by selectively enabling or disabling parts of it.
3. **Clarification**: Comments can clarify complex or obscure sections of code, making it easier for other developers (or even the original author) to understand the purpose or functionality of specific code segments. This is particula r ly helpful in large or complex codebases.

TYPES OF COMMENTS:

1. **Single-line comments:** These start with two forward slashes (//) and continue until the end of the line.

For example:

```
// This is a single-line comment

// This is a single-line comment
$variable = 42; // Assigning a value to a variable
```

2. **Multi-line comments:** These start with `/*` and end with `*/`, allowing you to span multiple lines.

For example:

```
/*
This is a multi-line comment
that spans multiple lines
*/

/**
 * Calculates the sum of two numbers.
 *
 * @param int $a The first number.
 * @param int $b The second number.
 * @return int The sum of $a and $b.
 */
function add($a, $b) {
    return $a + $b;
}
```

SENDING DATA TO THE WEB BROWSER:

Sending data to the web browser in PHP typically involves using functions like **echo**, **print**, or **printf** to output content directly to the webpage. In PHP, sending data to the web browser involves generating content dynamically on the server and then sending it to the client's web browser for display. This process typically involves using PHP code embedded within HTML or directly sending HTTP headers and content from PHP scripts.

Here's a step-by-step explanation of how data is sent to the web browser in PHP

1) **Server-side Processing**: When a client requests a PHP page from a web server, the server executes the PHP code contained within the requested file. This PHP code can generate HTML, CSS, JavaScript, or any other type of content dynamically based on the logic defined in the script.
2) **Generating Content**: Within the PHP script, you can use various PHP constructs and functions to generate the desired content. This content may include HTML markup, database query results, dynamically generated images, or any other type of data that needs to be displayed to the user.
3) **Outputting Content**: Once the content is generated, you can output it to the web browser using PHP's output functions such as **echo**, **print**, **printf**, or **print_r**. These functions send the content directly to the output buffer, which will eventually be sent to the client's browser.

4) **Headers** : In addition to the main content, you can also send HTTP headers using the **header** function. Headers are used to provide additional information to the browser, such as the content type (**Content-Type**), caching instructions, redirection, and more. Headers must be sent before any actual output is sent to the browser, otherwise, you'll encounter errors.

5) **HTTP Response**: Once all the content and headers have been generated, PHP sends the HTTP response to the web server, which then delivers it to the client's web browser over the network.

6) **Client-side Rendering**: The web browser receives the HTTP response from the server and renders the content according to the HTML, CSS, and JavaScript it contains. Any dynamic content generated by PHP will be integrated seamlessly into the final web page displayed to the user.

To send data to the web browser using several methods, including:

1.echo: The **echo** statement is used to output data directly to the web browser. It can output strings, variables, or a combination of both.

For example:

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

2.print: Similar to **echo**, the **print** statement is used to output data to the browser. It can also output strings and variables.

For example:

```
<?php
$message = "Hello, world!";
print $message;
?>
```

3.print_r: The **print_r** function is used to display information about a variable in a more readable format. It is commonly used for debugging purposes.

For example:

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

OUTPUT:

```
Array ( [0] => apple [1] => banana [2] => orange )
```

4.var_dump: The **var_dump** function is similar to **print_r**, but it provides more detailed information about a variable, including its type and value. It is also commonly used for debugging.

For example

```php
<?php
$array = array("apple", "banana", "orange");
var_dump($array);
?>
```

Output:

```
array(3) { [0]=> string(5) "apple" [1]=> string(6) "banana" [2]=> string(6) "orange" }
```

5.Headers: You can also send HTTP headers to the browser using the **header** function. Headers are used to provide additional information to the browser, such as content type, caching instructions, or redirection.

For example:

```php
<?php
header("Content-Type: text/plain");
echo "This is plain text";

?>
```

DATA TYPES IN PHP:

- The data type is a fundamental aspect of a programming language, defining what kind of data can be stored and manipulated in a variable or expression
- In PHP, data refers to information that is stored, processed, and manipulated within a PHP script. This data can come from various sources such as user input, databases, files, or generated within the script itself.

Uses Data Types Using Php

1.Data Validation: By defining data types, you can ensure that the input or stored data meets certain criteria. For example, you can ensure that a variable intended to hold an integer value does not accidentally receive a string.

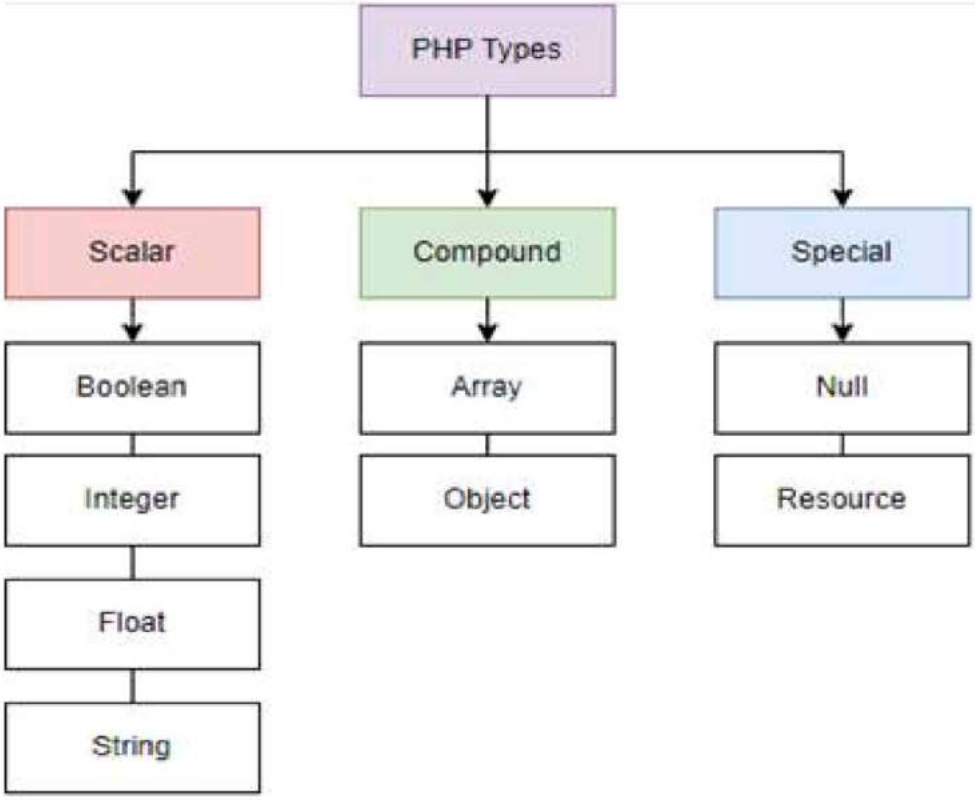
2. Memory Allocation: PHP allocates memory differently based on the data type. For example, integers require less memory than strings. By understanding data types, you can optimize memory usage in your PHP applications.

3. Function Behaviour: Some PHP functions behave differently based on the data types of their arguments. For instance, mathematical functions may only accept numeric values.

4. **Database Interactions:** When interacting with databases, it's essential to handle data types correctly to ensure data integrity and prevent SQL injection attacks.

5. **Output Formatting:** Data types can influence how data is formatted and displayed to users. For instance, formatting a date as a string for user-friendly presentation.

DATA TYPES IN PHP



1.Scalar Data types:

In PHP, a scalar data type refers to a data type that can hold only a single value at a time.
Scalar data types in PHP are essential for storing and manipulating individual values in variables.
These data types represent individual values and cannot be further subdivided.
PHP has four scalar data types:
1.Boolean:

- Hold only two values, either TRUE or FALSE.
- Successful events will return true and unsuccessful events return false.
- NULL type values are also treated as false in Boolean.
- If a string is empty then it is also considered as false in booleandata type.

Example:

```
<?php
```

```
    if(TRUE)
        echo "This condition is
    TRUE";if(FALSE)
        echo "This condition is not TRUE";
?>
```

Output: This condition is TRUE
This condition is not TRUE

Integer:

- Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point.
- They can be decimal (base 10), octal (base 8) or hexadecimal (base 16).
- The default base is decimal (base 10).
- The octal integers can be declared with leading 0 and thehexadecimal can be declared with leading 0x.
- The range of integers must lie between -2^31 to 2^31.

Example:

```
<?php
    // decimal base integers
    $deci1 = 50;
    $deci2 = 654;
    $sum=$deci1 + $deci2;
    echo $sum;
 ?>
```

Output: 704

Float(or)double:

- It can hold numbers containing fractional or decimal part including positive and negative numbers.
- By default, the variables add a minimum number of decimalplaces.

Example:

```
<?php
    $val1 = 50.85;
    $val2 = 654.26;
    $sum = $val1 + $val2;

    echo $sum;
 ?>
```

Output: 705.11

String:

- ❖ It can hold letters or any alphabets, even numbers are included.
- ❖ These are written within double quotes during declaration.
- ❖ The strings can also be written within single quotes but it will be treated differently while printing variables.

Example:

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

Output:

Hello JSSCOLLEG

Hello $college

Example:

```
<?php
        if(TRUE)
                echo "This condition is
        TRUE";if(FALSE)
                echo "This condition is not TRUE";
?>
```

Output: This condition is TRUE
This condition is not TRUE

Integer:

- Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point.
- They can be decimal (base 10), octal (base 8) or hexadecimal (base 16).
- The default base is decimal (base 10).
- The octal integers can be declared with leading 0 and thehexadecimal can be declared with leading 0x.
- The range of integers must lie between -2^31 to 2^31.

Example:

```
<?php
        // decimal base integers
        $deci1 = 50;
        $deci2 = 654;
        $sum=$deci1 + $deci2;
        echo $sum;
 ?>
```

Output: 704

Float(or)double:

- It can hold numbers containing fractional or decimal part including positive and negative numbers.
- By default, the variables add a minimum number of decimalplaces.

Example:

```
<?php
        $val1 = 50.85;
        $val2 = 654.26;
        $sum = $val1 + $val2;
        echo $sum;
 ?>
```

Output: 705.11

String:

- ❖ It can hold letters or any alphabets, even numbers are included.
- ❖ These are written within double quotes during declaration.
- ❖ The strings can also be written within single quotes but it will be treated differently while printing variables.

Example:

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

Output:

Hello JSSCOLLEG
Hello $college

Example:

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

```
        echo "This condition is not TRUE";
?>
```

Output: This condition is TRUE

This condition is not TRUE

Integer:

- Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point.
- They can be decimal (base 10), octal (base 8) or hexadecimal (base 16).
- The default base is decimal (base 10).
- The octal integers can be declared with leading 0 and the

  hexadecimal can be declared with leading 0x.
- The range of integers must lie between -2^31 to 2^31.

Example:

```
<?php
    // decimal base integers
    $deci1 = 50;
    $deci2 = 654;
    $sum=$deci1 + $deci2;
    echo $sum;
?>
```

Output: 704

Float(or)double:

- It can hold numbers containing fractional or decimal part including positive and negative numbers.
- By default, the variables add a minimum number of decimalplaces.

Example:

```
<?php
```

```
    $val1 = 50.85;
    $val2 = 654.26;
    $sum = $val1 + $val2;

    echo $sum;
?>
```

Output: 705.11

String:

- ❖ It can hold letters or any alphabets, even numbers are included.
- ❖ These are written within double quotes during declaration.
- ❖ The strings can also be written within single quotes but it will be treated differently while printing variables.

Example:

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

Output:

Hello JSSCOLLEG
Hello $college

2.Compound Data Type:

In PHP, compound data types are those that can hold multiple values or a collection of values. Compound data types are versatile and powerful constructs in PHP that allow you to work with complex data structures and implement advanced programming concepts like data abstraction, encapsulation, and polymorphism.These data types allow you to organize and manipulate related data efficiently.

1.ARRAY:

- Array is a compound data-type which can store multiple values ofsame data type.
- An array is a collection of key-value pairs, where each value is associated with a unique key.
- Arrays can hold elements of different data types, such as integers, floats, strings, or even other arrays.
- Arrays in PHP can be indexed (numerically or associative) or non-indexed.

Example1:

```
<?php
        $intArray = array( 10, 20 , 30);
        echo "First Element: $intArray[0]\n";
        echo "Second Element:
        $intArray[1]\n";
        echo "Third Element: $intArray[2]\n";
?>
```

Output: First Element: 10
Second Element: 20
Third Element: 30
OR

Example2:

```
<?php
$fruits = array("apple", "banana", "orange");
echo $fruits[0];
?>
// Output: apple
```

2.OBJECT: In PHP, an object is a compound data type that allows you to create instances of classes. Objects encapsulate both data (properties or attributes) and behavior (methods or functions) related to a specific entity or concept.

Objects are the instances of user-defined classes that can store both values and functions.

They must be explicitly declared.

Example1:

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

```
  $obj = new bike();
  $obj -> model();
?>
```

Output:

Bike Model: Royal Enfield

Example2:

```
<?php
class Person {
   public $name;
   public function greet() {
      return "Hello, my name is {$this->name}.";
   }
}

$personObj = new Person();
$personObj->name = "Madesh";
echo $personObj->greet();
?>
```

Output: Hello, my name is Madesh.

3.SPECIAL DATA TYPE: These special data types provide additional functionality and flexibility to PHP developers, allowing them to address a wide range of programming requirements and scenarios effectively.

1.Resources: Resources are special variables holding references to external resources, such as file handles or database connections. They are created and managed by PHP extensions or external libraries and enable interaction with external entities.

Example:

```
// Open a file resource
$file = fopen("example.txt", "r");

// Read from the file

echo fread($file, filesize("example.txt"));

// Close the file resource

fclose($file);
```

2. NULL: NULL represents a variable with no value. It is commonly used to indicate the absence of a value or to reset a variable. NULL is a special value in PHP that represents a variable with no value. It is often used to

indicate the absence of a value or to reset a variable to an undefined state.

Example: $var = NULL; // Variable with no value assigne

KEYWORDS IN PHP:

In PHP, keywords are predefined reserved words that have special meanings and cannot be used for naming variables, functions, classes, or other identifiers.

Understanding and using these keywords effectively is essential for writing clean, efficient, and maintainable PHP code.

Some key rules for using keywords in PHP include:

1. **Reserved Words**: Keywords are reserved for specific purposes and cannot be used as identifiers in your code. For example, you cannot create a variable named `if` or a function named `echo`.
2. **Case Insensitivity**: PHP keywords are case-insensitive. This means that you can use them in any case (e.g., `if`, `IF`, `If`) and they will have the same meaning. However, it's a good practice to use lowercase for consistency.
3. **Not All Keywords Are Reserved**: Not all words that are part of PHP's syntax are reserved keywords. For example, `array` is a language construct but not a reserved keyword, so you can use it as a variable name.
4. **Global Scope**: Keywords are global in scope, meaning they have the same meaning throughout your PHP script or application.
5. **Documentation Reference**: PHP documentation usually highlights keywords with special formatting or styling to distinguish them from other elements.

THE MAIN TYPES OF KEYWORDS IN PHP:

1. **Control Structures**: Keywords like **if, else, elseif, switch, case, default, while, do, for, foreach, break, continue, and return** are used to control the flow of execution in a PHP script. They enable developers to make decisions, iterate over data, and manage program flow.

```
Example: $age = 20;
if ($age >= 18) {
   echo "You are an adult.";
} else {
   echo "You are not yet an adult.";
}
```

2. **Functions and Classes**: Keywords such as **function, class, extends, implements, interface, abstract, final, new,** and instanceof are used to define functions and classes in PHP. They allow developers to encapsulate code into reusable modules and create object-oriented structures.

```
Example: function greet($name) {
   echo "Hello, $name!";
}
greet("Madesh");
```

3. **Visibility Modifiers**: Keywords like **public, protected, and private** are used within classes to specify the visibility of properties and methods. They control access to class members from within the class itself and from external code.

```
Example: class Car {
   public $brand;
   public $model;

   function __construct($brand, $model) {
      $this->brand = $brand;
      $this->model = $model;
   }

   function getInfo() {
      return "$this->brand $this->model";
   }
}
$car = new Car("Toyota", "Camry");
```

```
echo $car->getInfo(); // Output: Toyota Camry
```

4. **Error Handling**: Keywords such as **try, catch, finally, and throw are used** for exception handling in PHP. They allow developers to gracefully handle errors and exceptions that occur during script execution.

   Example:
   ```
   try {
       // Code that may throw an exception
       $result = 10 / 0;
   } catch (Exception $e) {
       // Handle the exception
       echo "An error occurred: " . $e->getMessage();
   }
   ```

5. **Include and Require**: Keywords like include, **include_once, require, and require_once** are used to include and require external PHP files into the current script. They enable code reuse and modularization by allowing developers to split their code into separate files.
6. **Constants**: Keywords like **const** and **define** are used to define constants in PHP. Constants are immutable values that can be referenced throughout a script and provide a way to store and reuse fixed values.
7. **Namespace**: Keywords such as **namespace** and **use** are used to define and import namespaces in PHP. Namespaces help organize code into logical groupings and prevent naming conflicts between different parts of a PHP application.
8. **Variable Scope**: Keywords like **global** and **static** are used to define variable scope in PHP. They determine the visibility and lifetime of variables within different parts of a script or application.
9. **Type Declaration**: Keywords like **bool, int, float, string, array, object, resource, callable, iterable, and void** are used for type declaration in PHP. They allow developers to specify the data type of variables, parameters, and return values, which can help improve code clarity and maintainability.

Keyword	Description	fn	Declare an arrow function
abstract	Declare a class as abstract	for	Create a for loop
and	A logical operator	foreach	Create a foreach loop
as	Used in the foreach loop	function	Create a function
break	Break out of loops and switch statements	global	Import variables from the global scope
callable	A data type which can be executed as a function	goto	Jump to a line of code
case	Used in the switch conditional	if	Create a conditional statement
catch	Used in the try..catch statement	implements	Implement an interface
class	Declare a class	include	Embed code from another file
clone	Create a copy of an object	include_once	Embed code from another file
const	Define a class constant	instanceof	Test an object's class
continue	Jump to the next iteration of a loop	insteadof	Resolve conflicts with traits
declare	Set directives for a block of code	interface	Declare an interface
default	Used in the switch statement	isset	Check if a variable exists and is not null
do	Create a do...while loop	list	Assigns array elements into variables
echo	Output text	namespace	Declares a namespace
else	Used in conditional statements	new	Creates an object
elseif	Used in conditional statements	or	A logical operator
empty	Check if an expression is empty	print	Output text
enddeclare	End a declare block	private	Declare a property, method or constant as private

endfor	End a for block	protected	Declare a property, method or constant as protected
endforeach	End a foreach block	public	Declare a property, method or constant as public
endif	End an if or elseif block	require	Embed code from another file
endswitch	End a switch block	require_once	Embed code from another file
endwhile	End a while block	return	Exit a function and return a value
extends	Extends a class or interface	static	Declare a property or method as static
final	Declare a class, property or method as final	switch	Create a switch block
finally	Used in the try...catch statement	throw	Throw an exception
fn	Declare an arrow function	trait	Declare a trait
for	Create a for loop	try	Create a try...catch structure
foreach	Create a foreach loop	unset	Delete a variable or array element
function	Create a function	use	Use a namespace
global	Import variables from the global scope	var	Declare a variable
goto	Jump to a line of code	while	Create a while loop or end a do...while loop
if	Create a conditional statement	xor	A logical operator
implements	Implement an interface	yield	Used in generator functions
include	Embed code from another file		

USING VARIABLES:

- In PHP, a variable is a symbolic name that represents a value. Variables are used to store and manipulate data within a script.
- They can hold various types of data, such as numbers, strings, arrays, objects, or even resource handles.
- Variables in PHP are declared using the dollar sign ($) followed by the variable name. Variable names must start with a letter or underscore, followed by any combination of letters, numbers, or underscores.
- In PHP, a variable is declared using a **$ sign** followed by the variable name. Here, some important points to know about variables.

Variables in PHP have several important uses in programming:

1. **Storage**: Variables are used to store data temporarily during the execution of a script. This allows values to be accessed and manipulated throughout the program.
2. **Data Manipulation**: Variables can hold various types of data, including numbers, strings, arrays, objects, and resource handles. They enable developers to perform operations, calculations, and transformations on this data.
3. **Dynamic Content**: Variables are commonly used to generate dynamic content in web applications. For example, they can be used to display user input, database query results, or computed values on a webpage.
4. **Control Structures**: Variables are often used in control structures such as loops and conditional statements to control the flow of execution based on conditions and iterate over data structures.
5. **Passing Values**: Variables allow values to be passed between different parts of a script, such as between functions, classes, or included files. This facilitates modular and reusable code.
6. **Configuration**: Variables can be used to store configuration settings and parameters, making it easy to customize the behavior of a script without modifying its code directly.
7. **Scope Management**: PHP supports different scopes for variables, including global, local, static, and class-level scopes. Variables help manage data visibility and accessibility within different parts of a script.

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax :
$variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

In PHP, variables can be classified into different types based on their scope and behavior. Here are the main types of variables in PHP:

1.Local Variables: These are variables declared within a function or a code block and are only accessible within that function or block.

Example:

```
function myFunction() {
   $localVariable = "This is a local variable.";
   // $localVariable is only accessible within myFunction()
}
```

2. Global Variables: These are variables declared outside of any function or code block and are accessible from anywhere in the script, including inside functions.

Example:

```
$globalVariable = "This is a global variable.";
function myFunction() {
   global $globalVariable;
   echo $globalVariable;
}
```

3. Static Variables: These are variables that retain their value between function calls. They are initialized only once when the function is first called and then retain their value between subsequent calls.

```
Example:function counter() {
   static $count = 0;
   $count++;
   echo $count;
}
counter(); // Output: 1
counter(); // Output: 2
```

4. Superglobal Variables: These are predefined variables provided by PHP that are accessible from any part of the script. They include variables like $_GET, $_POST, $_SESSION, $_COOKIE, $_SERVER, etc., and are commonly used to collect form data, manage sessions, handle cookies, etc.

Example: $name = $_GET['name']; // Accessing data from the URL query string

5.Variable Variables: These are variables whose names can be dynamically determined at runtime based on the value of another variable.

```
Example: $variableName = 'example';
$$variableName = 'Value assigned dynamically';
echo $example; // Output: Value assigned dynamically
```

CONSTANTS IN PHP:

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

PHP constants can be defined by 2 ways:

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

- Constants are similar to the variable except once they defined they can never be undefined or changed. They remain constant across the entire program.
- In PHP, constants are like variables but once they are defined, they cannot be changed or undefined.
- They are useful for storing values that remain constant throughout the execution of a script. Constants are defined using the `define()` function or the `const` keyword.

Note: PHP constants follow the same PHP variable rules.

- it can be started with a letter or underscore only.
- Conventionally, PHP constants should be defined in uppercase letters.
- Unlike variables, constants are automatically global throughout the script.

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

Syntax: define(name, value, case-insensitive)

name: It specifies the constant name.

value: It specifies the constant value.

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

The example to define PHP constant using define().

```
<?php
define("MESSAGE","JSS COLLEGE");
echo MESSAGE;
?>
```

Output:

JSS COLLEGE

Create a constant with **case-insensitive** name

2. constant keyword: The const keyword in PHP is used to define class constants. Class constants, unlike class variables, remain constant and cannot be changed or redefined after they are declared. Here's how you can use const keyword in PHP

Syntax: CONSTANT_NAME = "constant_value";

Example:

```
<?php
// Define constants using const keyword
const PI = 3.14;
const GREETING = "Hello, world!";
// Accessing constants
echo "The value of PI is: " . PI . "<br>";
echo "The greeting message is: " . GREETING . "<br>";
?>
```

Output:

The value of PI is: 3.14

The greeting message is: Hello, world!

Example3:

```
<?php
// Define constants using define() function
define("PI", 3.14);
define("GREETING", "Hello, world!");
// Define constants using const keyword (since PHP 5.3)
const SITE_NAME = "My Website";
const VERSION = "1.0";
// Accessing constants
echo "The value of PI is: " . PI . "<br>";
echo "The greeting message is: " . GREETING . "<br>";
echo "Welcome to " . SITE_NAME . ", Version " . VERSION . "<br>";
?>
```

OUTPUT: The value of PI is: 3.14

The greeting message is: Hello, world!

Welcome to My Website, Version 10

DIFFERENCE BETWEEN CONSTANTS AND VARIABLES

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

EXPRESSIONS IN PHP: A PHP expression is a combination of variables, operators, and functions that evaluates to a single value. It can be as simple as a variable name or as complex as a mathematical formula or function call.

Some common uses of expressions in PHP

1. **Data Manipulation**: Expressions are used to manipulate data, perform calculations, and transform values. This includes arithmetic operations, string manipulation, and formatting data.
2. **Conditional Logic**: Expressions are essential for implementing conditional logic, such as if statements, switch statements, and ternary operators. They allow developers to control the flow of execution based on specified conditions.
3. **Looping Constructs**: Expressions are used in looping constructs like for loops, while loops, and foreach loops to iterate over arrays, perform repetitive tasks, and process collections of data.

4. **Function Invocation**: Expressions are used to invoke functions and methods, passing arguments and parameters as needed. This allows developers to encapsulate functionality and reuse code throughout their applications.
5. **Error Handling**: Expressions are used for error handling and exception handling, including evaluating conditions to trigger error messages, logging errors, and handling exceptional cases gracefully.

TYPES OF EXPRESSION IN PHP

1. **Arithmetic Expressions**
2. **String Expression**
3. **Compaision Expression**
4. **Logical Expression**
5. **Assignment Expression**
6. **Ternary Expression**
7. **Array Expression**
8. **Function call expression**

1.Arithmetic Expressions: These expressions involve mathematical operations like addition, subtraction, multiplication, and division.

Example:

```
<?php
$a = 10;
$b = 5;
$result = $a + $b * 2; // Arithmetic expression
echo "Result of arithmetic expression: $result";
?>
```

2. String Expressions: These expressions involve operations on strings, such as concatenation.

Example:

```
<?php
$greeting = "Hello, " . "world!"; // String expression
echo $greeting;
?>
```

3. **Comparison Expressions**: These expressions compare values and return a boolean result (`true` or `false`).

Example:

```
<?php
$x = 10;
$y = 20;
$is_greater = ($x > $y); // Comparison expression
echo "Is x greater than y? " . ($is_greater ? "Yes" : "No");
?>
```

4. Logical Expressions: These expressions involve logical operations such as AND (&&), OR (||), and NOT (!).

Example:

```
<?php
$age = 25;
$is_adult = ($age >= 18 && $age <= 65); // Logical expression
echo "Is the person an adult? " . ($is_adult ? "Yes" : "No");
?>
```

5.Assignment Expressions: These expressions involve assigning values to variables.

Example:

```
<?php
$num = 10;
$num += 5; // Assignment expression
echo "Value of num after assignment: $num";
?>
```

6. Ternary Expressions: These are conditional expressions that evaluate to one value if a condition is true and another if the condition is false.

Example:

```
<?php
$score = 85;
$result = ($score >= 60) ? "Pass" : "Fail"; // Ternary expression
echo "Result: $result";
?>
```

7. Array Expressions: These expressions involve operations on arrays, such as accessing elements by index.

Example:

```
<?php
$fruits = ["Apple", "Banana", "Orange"]; // Array expression
echo "First fruit: " . $fruits[0];
?>
```

8. Function Call Expressions: These expressions involve calling functions and passing arguments.

Example:

```
<?php
$string = "Hello, world!";
$length = strlen($string); // Function call expression
echo "Length of the string: $length";
?>
```

OPERATORS IN PHP.

- PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values.
- They are used to manipulate and compare values, control program flow, and perform various other tasks.
- In programming, an "operator" is a symbol or keyword that performs an operation on one or more "operands."
- Operands are the values or variables that an operator acts upon.

  Example, in the expression `3 + 5`, the `+` symbol is the operator, and `3` and `5` are the operands.

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

1. **Unary Operators:** works on single operands such as ++, -- etc.
2. **Binary Operators:** works on two operands such as binary +, -, *, / etc.
3. **Ternary Operators:** works on three operands such as "?:".

PHP Operators can be categorized in following forms

1. Arithmetic Operators
2. Assignment Operators
3. Bitwise Operators
4. Comparison Operators
5. Incrementing/Decrementing Operators
6. Logical Operators

7. String Operators
8. Array Operators
9. Type Operators
10. Execution Operators
11. Error Control Operators

1. Arithmetic Operators:

Arithmetic operators are used for mathematical calculations. The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

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

Example: <?php

$a = 10;

$b = 5;

// Addition

$result = $a + $b; // $result = 15

// Subtraction

$result = $a - $b; // $result = 5

// Multiplication

```
$result = $a * $b; // $result = 50

// Division

$result = $a / $b; // $result = 2

// Modulus

$result = $a % $b; // $result = 0 (remainder of 10 / 5)

?>
```

2. Assignment Operators:

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

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

Example:

```
<?php

$a = 10;

$b = 5;

// Compound assignment

$a += $b; // Equivalent to: $a = $a + $b; $a = 15

?>
```

3. Bitwise Operators: The bitwise operators are used to perform bit-level operations on operands.

- These operators allow the evaluation and manipulation of specific bits within the integer.
- Bitwise operators perform operations at the bit level.

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

Example:

```
<?php
$a = 5; // 101
$b = 3; // 011
// Bitwise AND
$result = $a & $b; // $result = 1 (001)
// Bitwise OR
$result = $a | $b; // $result = 7 (111)
// Bitwise XOR
$result = $a ^ $b; // $result = 6 (110)
// Bitwise NOT
$result = ~$a; // $result = -6 (not 010)
?>
```

4. Comparison Operators:

Comparison operators compare two values and return a Boolean result. Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	**Spaceship**	**$a <=>$b**	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

Example:
```
<?php
$a = 10;
$b = 5;

// Equal to
$result = $a == $b; // false

// Not equal to
$result = $a != $b; // true

// Greater than
$result = $a > $b; // true
```

```
// Less than or equal to
$result = $a <= $b; // false
?>
```

5. Incrementing/Decrementing Operators:

These operators are used to increase or decrease the value of variables.

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

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

Example: <?php

$a = 10;

// Pre-increment

++$a; // $a = 11

// Post-decrement

$b = $a--; // $b = 11, $a = 10

?>

6. Logical Operators:

Logical operators perform logical operations on Boolean values.

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

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

Example:

```
<?php
$a = true;
$b = false;
// Logical AND
$result = $a && $b; // false
// Logical OR
$result = $a || $b; // true
// Logical NOT
$result = !$a; // false
?>
```

7. String Operators:

String operators are used for concatenating strings. The string operators are used to perform the operation on strings. There are two string operators in PHP.

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

Example:<?php

$a = "Hello";

$b = "World";

// Concatenation

$result = $a . $b; // $result = "HelloWorld"
?>

8. Array Operators:

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

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

Example:

<?php

$array1 = array("a" => "apple", "b" => "banana");

$array2 = array("b" => "blueberry", "c" => "cherry");

// Union

$result = $array1 + $array2; // $result = ["a" => "apple", "b" => "banana", "c" => "cherry"]

?>

9. Type Operators:

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

Example:

```
<?php
class MyClass {}
$obj = new MyClass();
// instanceof
$result = $obj instanceof MyClass; // true
?>
```

10. Execution Operators:

Execution operators execute commands as if they were executed directly from the command line. PHP has an execution operator **backticks (``)**. PHP executes the content of backticks as a shell command. Execution operator and **shell_exec()** give the same result.

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

Example:

```
<?php
// Execute command and return output
$output = `ls -l`;
?>
```

11. Error Control Operators:

Error control operators are used to suppress errors or change error reporting settings. PHP has one error control operator, i.e., **at (@) symbol**. Whenever it is used with an expression, any error message will be ignored that might be generated by that expression.

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

Example: <?php

// Suppress errors

$result = @file_get_contents('nonexistent_file.txt');

// Set error reporting level

error_reporting(E_ALL & ~E_NOTICE);

?>

PHP OPERATORS PRECEDENCE

Operators	Additional Information	Associativity
clone new	clone and new	non-associative
[	array()	left
**	arithmetic	right
++ -- ~ (int) (float) (string) (array) (object) (bool) @	increment/decrement and types	right
instanceof	types	non-associative
!	logical (negation)	right
* / %	arithmetic	left
+ - .	arithmetic and string concatenation	left
<< >>	bitwise (shift)	left
< <= > >=	comparison	non-associative
== != === !== <>	comparison	non-associative
&	bitwise AND	left

^	bitwise XOR	left
|	bitwise OR	left
&&	logical AND	left
||	logical OR	left
?:	ternary	left
= += -= *= **= /= .= %= &= |= ^= <<= >>= =>	assignment	right
and	logical	left
xor	logical	left
or	logical	left
,	many uses (comma)	left