Unit 1 :

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, Comments in PHP, Sending Data to the Web Browser, Data types, Keywords, Variables, Constants in PHP, Expressions in PHP, Operators in PHP. **Conditional statements:** if, if-else, switch, The? Operator**, Looping statements:** while Loop, do-while Loop, for Loop, foreach loop, break, continue.

Xampp Download Link : https://www.apachefriends.org/downloa...
Visual Studio Code Download Link : https://code.visualstudio.com/download

What You Should Already Know

Before you continue you should have a basic understanding of the following:

• HTML
• CSS
• JavaScript

What is PHP?

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

What is a PHP File?

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

What Can PHP Do?

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

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

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Introduction to PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).

PHP was created by Rasmus Lerdorf in 1994 but appeared in the market in 1995. PHP 7.4.0 is the latest version of PHP, which was released on 28 November. Some important points need to be noticed about PHP are as followed:

Some facts about PHP:

- PHP has many syntaxes similar to C, Java, and Perl, and has many unique features and specific functions.
- PHP page is a file with a .php extension can contain a combination of HTML Tags and PHP scripts.
- PHP is Server-side scripting language: Server-side scripting means that the PHP code is processed on the web server rather than the client machine.
- PHP supports many databases (MySQL and PHP combination is widely used).
- PHP is an open-source scripting language.
- PHP is an object-oriented language.
- PHP is free to download and use.
- PHP is simple and easy to learn language.

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 ihe website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. For example - Registration form.
- PHP 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, IMA P, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first.
- 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. PH P can develop dynamic websites easily.
however you must have the basic the knowledge of the 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 wake 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 the data with a webserver.
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.

History and Features of 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 (born 22 November 1965) is a programmer He co-authored and inspired the PHP scripting language, authoring the first two versions of the language and participated in the development of later versions led by a group of developers including Jim Winstead (who later created blog), Stig Bakken, Shane Caraveo, Andi Gtitrnuns, and Zeev Shrunki. 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 applications.
- 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.
  - 1994: PHP/FI (Forms Interpreter) - First version
  - 1997: PHP/FI 2.0 - Rewritten with better functionality
  - 1998: PHP 3.0 - Major rewrite, became popular
  - 2000: PHP 4.0 - Introduced Zend Engine
  - 2004: PHP 5.0 - Object-oriented programming support
  - 2015: PHP 7.0 - Major performance improvements
  - 2020: PHP 8.0 - Latest with JIT compiler

1. **PHP** 1.0:It First version was released version 1.0.
2. **PHP** 2.0:In April 1996. Rasmus introduced PHP/Fl. included built-in support for DBM, MySQL, and Postgres95 databases, cookies' user-defined function support. PHP/FI was given the **version 2.0** status.
3. PHP3.0: Hypertext Pre-processor — PHP 3.0 version came about when Zeev Stiraski and Audi 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
4. 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
5. **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 development team includes dozens of developers and others working on PHP-related and supporting projects such as PEAR, PECL, and documentation.
6. **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

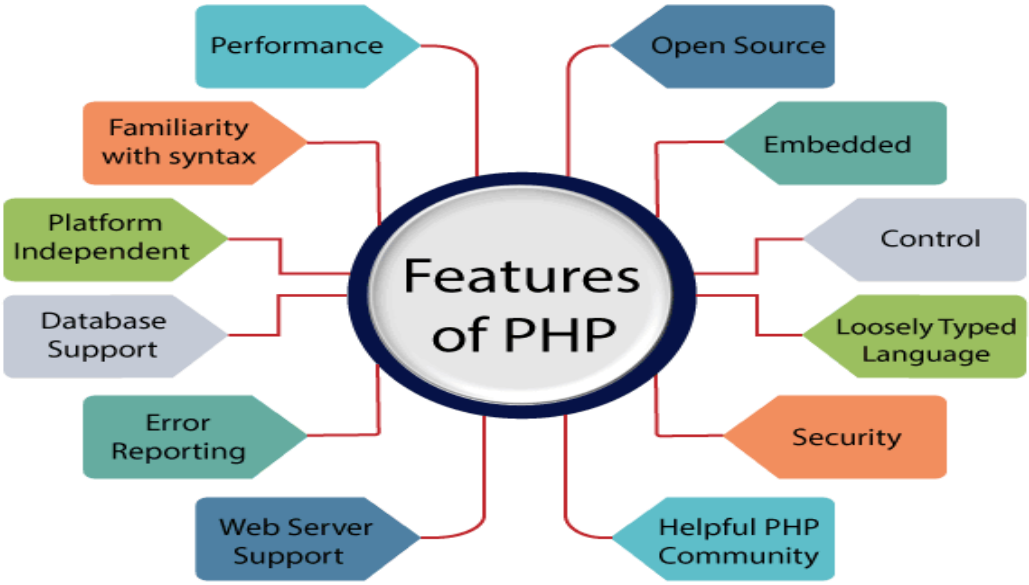
Type Declarations and Anonymous Classes.

7. **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 8.2.1, released on July 4th, 2023.
8. **PHP 8.2.3 :** It is the latest version of php released in the year of 18"Jan 2024.

UNDERSTANDING PHP:

FEATURES OF PHP

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



Feature	Description	Benefit
Open Source	Free to use and modify	Cost-effective development
Cross-Platform	Works on Windows, Linux, macOS	High compatibility
Server-Side Scripting	Executes on web server	Dynamic web content
Database Integration	Supports MySQL, PostgreSQL, Oracle	Flexible database connectivity
Easy Learning Curve	Simple syntax similar to C/C++	Quick development process
Large Community	Extensive documentation and support	Rich ecosystem

1. 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.
2. 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.
3. Familiarity with syntax: PHP has easily understandable syntax. Programmers are comfortable coding with it.
4. Embedded: PHP code can be easily embedded within HTML tags and script.
5. 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.
6. Database Support: PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
7. 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.
8. 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.
9. Web servers Support: PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
10. Security: PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
11. 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.
12. 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.

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 clients browser. This allows for dynamic content generation and interaction with databases.

2. **Cross-platform compatibility:** PHP urns on various platform, 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 team 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, al bowing 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 generation, 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 language and its ecosystem through forums, online communities, and conferences.

Applications of PHP

PHP is widely used in various applications across web development. Solve 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 WooColcmerce (WordPress plugin), Magento, and OpenCart, are built using PHP. PHP enables developers to create robust and scalable online stores with features like product catalog management, 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 PlaWorms:** 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 us, 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 programming 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 behaviors. 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 framework may vary. It's essential to carefully evaluate third-party components and ensure they meet the project's requirements and security standards.

INSTALLATION & CONFIGURATION OF PHP

Installing PHP with XAMPP/WAMP

What is XAMPP?

XAMPP is a free, cross-platform web server solution stack package consisting of:

X - Cross-platform

A - Apache HTTP Server

M - MariaDB/MySQL Database

P - PHP Programming Language
P - Perl Programming Language
What is WAMP?
WAMP is a Windows-based web development environment:
W - Windows Operating System
A - Apache HTTP Server
M - MySQL Database
P - PHP Programming Language
Installation Benefits

Aspect	XAMPP	WAMP
Platform Support	Windows, Linux, macOS	Windows only
File Size	Larger (~150MB)	Smaller (~50MB)
Components	More comprehensive	Focused essentials
Ease of Use	Beginner-friendly	Very simple interface

Step 1: System Requirements
Before installing PHP, ensure the system meets the following requirements:

- Operating System: Windows 7 / 8 / 10 / 11
- Minimum RAM: 2 GB
- Disk Space: At least 1 GB free
- Web Browser: Chrome / Edge / Firefox

Step 2: Download XAMPP
PHP is commonly installed using **XAMPP**, which includes:

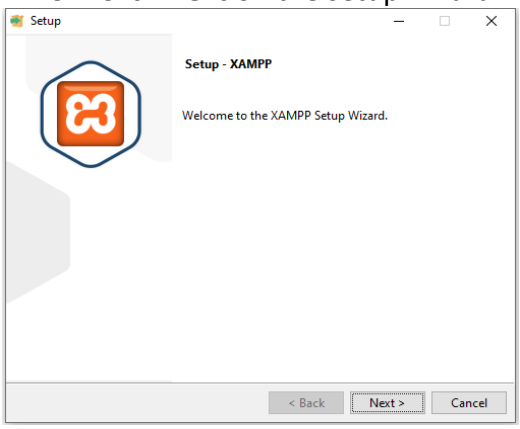
- Apache (Web Server)
- PHP
- MySQL

Steps:

1. Open a web browser
2. Go to https://www.apachefriends.org
3. Click **Download** for Windows version

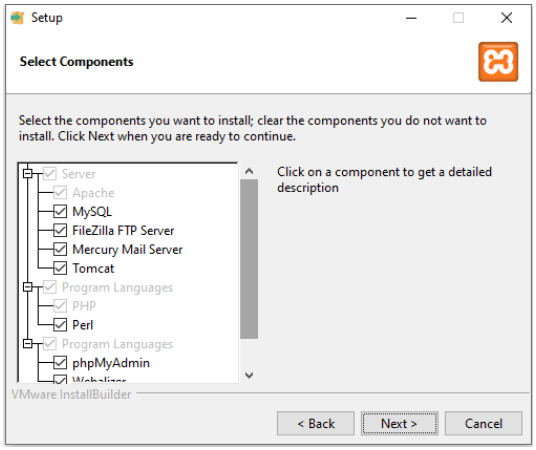
Step 3: Install XAMPP

1. Double-click the downloaded XAMPP installer
2. Click **Yes** if User Account Control appears
3. Click **Next** on the setup wizard



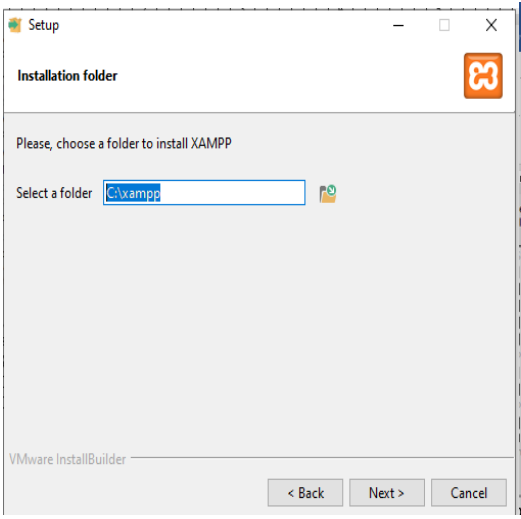
Step 4: Select Components

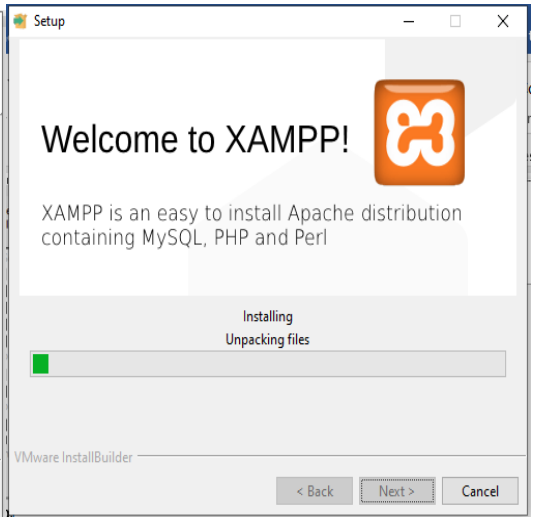
1. Ensure **Apache**, **MySQL**, and **PHP** are selected
2. Click **Next**



5: Choose Installation Folder

1. Select installation path (default recommended):
2. C:\xampp
3. Click **Next**



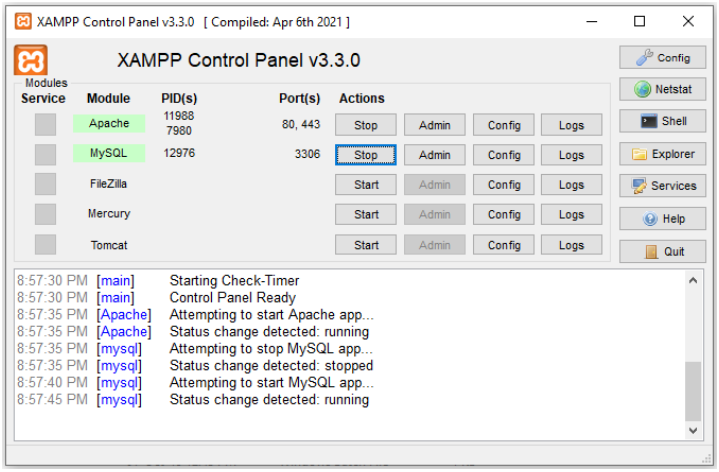


Step 6: Complete Installation

1. Click **Next** → **Finish**
2. XAMPP Control Panel opens automatically

Step 7: Start Apache Server

1. Open **XAMPP Control Panel**
2. Click **Start** next to **Apache**
3. Apache status turns **green**



Step 8: Verify PHP Installation

1. Open a browser
2. Type:
3. http://localhost
4. XAMPP dashboard appears

Step 9: Create PHP Test File

1. Go to:
2. C:\xampp\htdocs
3. Create a new file named:
4. test.php
5. Add the following code:

```
<?php
phpinfo();
?>
```

Step 10: Run PHP File

1. Open browser
2. Type:
3. http://localhost/test.php
4. PHP configuration page is displayed

Step 11: Configure PHP (Optional)

1. Open:
2. C:\xampp\php\php.ini
3. Modify settings such as:
    - upload_max_filesize
    - max_execution_time
4. Save file
5. Restart Apache server

Step 12: PHP Installation Successful

PHP is now successfully installed and configured on the system.

✓ Apache running
✓ PHP executing
✓ Localhost working

Installation Steps Overview 📋

1. **Download** the installer from official website
2. **Run** the installer with administrator privileges
3. **Select** components (Apache, MySQL, PHP, phpMyAdmin)
4. **Choose** installation directory
5. **Start** Apache and MySQL services
6. **Test** installation by accessing localhost
7. **Configure** virtual hosts if needed

Post-Installation Configuration

1. **Document Root**: Usually htdocs folder for web files
2. **Configuration Files**: httpd.conf for Apache, php.ini for PHP
3. **Port Settings**: Default HTTP port 80, MySQL port 3306
4. **Security**: Change default passwords, enable firewalls
5. **PHP Extensions**: Enable required extensions in php.ini

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";
?>
```

HTML relationship with PHP
When user wants to write a PHP code, he must know about the HTML without HTML user cannot create the webpage both HTML and PHP languages are important for web-development. The PHP code is embedded in HTML files.

Difference between HTML and PHP

HTML	PHP
HTML is a markup language used to create static WebPages.	PHP is an open source scripting language used for the development of dynamic Websites and dynamic web applications.
HTML is used for front end development or Client side.	PHP is used for back end development or Server side.
HTML is very easy to learn with less time it	PHP is also easy to learn but not as much as

Accepts some small mistakes and gives the output.	HTML it takes more time to learn and it won't accept the mistake in code.
HTML is compatible with almost all Browsers.	PHP also compatible with all types of Browsers.
HTML is used to organize the content of the websites. It focuses on how the contents need to display with different fonts, size and colors etc.	PHP is used to "interact" with the database to retrieve the information, storing the data, e-mail sending and provides the content to The HTML pages to display on the screen.
<html> <body> ---------- </body> </html>	<?PHP ?>

Whitespace in HTML and PHP:

Whitespace in HTML: In HTML, whitespace (spaces, tabs, line breaks) is generally insignificant. Multiple consecutive spaces are treated as a single space and line breaks are usually ignored.

This is a paragraph with multiple spaces and line break

White spaces in PHP: In PHP, white space can affect the readability of the code but is generally ignored during execution. However, in certain situations, like within strings, white space is preserved.

```
<?php
$greeting = "Hello          World!";
echo $greeting;
?>
```

Output:

Hello World!

PHP In HTML

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, including 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 fonts are used to collect user input, such as login credentials, registration details, search queries, of feedback. PHP scripts handle form submissions by accessing form data through the $_POST or $_GET super global 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 permissions. 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. PH P 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:** HTM L 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.

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.
2. **Separating Tokens :** white space is used to separate the tokens in PHP code such as variables, operators, function names, etc.
3. **Whitespace between function arguments:** Whitespace is used to separate arguments in function calls.
4. **Code Readability:** Whitespace is used to improve the readability of code by adding space between elements. Proper indentation and spacing wake 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 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 readability, 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 documentation 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.

Types of Whitespaces in PHP

- Space ( )
- Tab (\t)
- New line (\n)

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 four of communication between developers, helping to improve code readability and maintainability.

USES OF COMMENTS IN PHP

1. **Documentation:** Comments arc used to document code. providing explanations 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 Paris of it.
3. **Clarification:** Comments can clarity 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 particularly helpful in large or complex codebases.

TYPES OF COMMENTS:

1. **Single-line comments:**
   a. Using //

```
<?php
// This is a single-line comment
echo "Hello World";
?>
```

   b. Using #

```
<?php
# This is also a single-line comment
echo "Welcome to PHP";
?>
```

2. **Multi-line Comments**
   a. Using /* */

```
<?php
/*
This is a multi-line comment.
It can span multiple lines.
Used for explanations.
*/
```

```
echo "PHP Comments";
?>
```

3. **Inline Comments**
   Comments written on the same line as code.

```
<?php
echo "Hello"; // Prints Hello
?>
```

4. **Purpose of Comments**
   - Improve code readability
   - Help others understand the code
   - Useful for debugging
   - Helpful for documentation
5. **Comments are Ignored by PHP**
   PHP does **not execute comments**.

```
<?php
// echo "This will not be executed";
echo "This will be executed";
?>
```

6. **Best Practices**
   - Write clear and meaningful comments
   - Avoid unnecessary comments
   - Use comments for complex logic

TYPES OF ERRORS: Mistake or fault occurs in a program is called error. There are basically 4 types of errors in PHP:

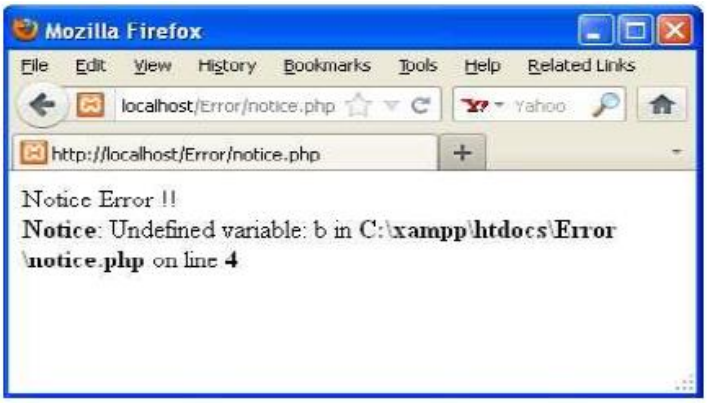
1. Notice
2. Warning
3. Fatal Error
4. Parse Error

1. NOTICE: Notice that an error is the same as a warning error i.e. in the notice error does not stop execution of the script. Notice that the error occurs when you try to access the undefined variable, and then produce a notice error.

Example

```
<?php
$a="Uday ";
echo "Notice Error !!";
echo $b;
?>
```

Output: In the above code we defined a variable which named $a. But we call another variable i.e. $b, which is not defined. So there will be a notice error produced but execution of the script does not
stop, you will see a message Notice Error !!. Like in the following
image:

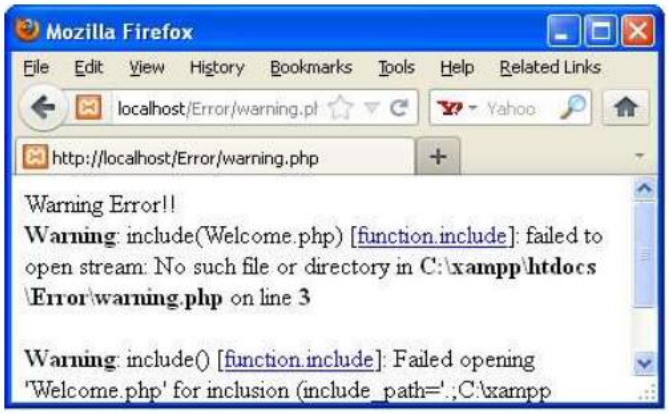


2. WARNING: Warning errors will not stop execution of the script. **The main reason for warning errors is to include a missing file or using the incorrect number of parameters in a function.**

Example:

```
<?php
echo "Warning Error!!";
include ("Welcome.php");
?>
```

Output: In the above code we include a welcome. Php file, however the welcome. Php file does not exist in the directory. So there will be a warning error produced but that does not stop the execution of the script i.e. you will see a message Warning Error !!.



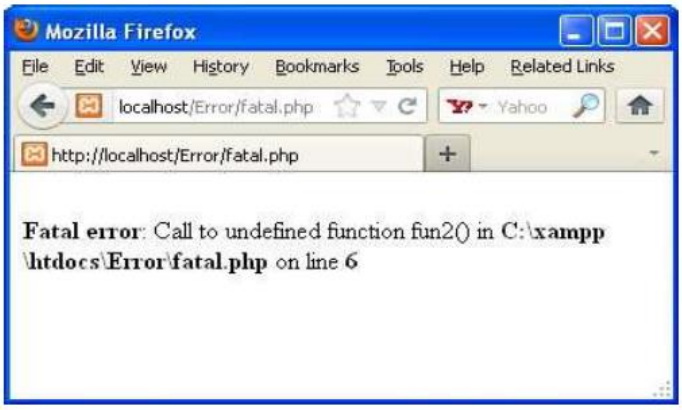
3. FATAL ERROR

Fatal errors are caused when PHP understands what you've written; however what you're asking it to do can't be done. Fatal errors stop the execution of the script. **If you are trying to access the undefined functions, then the output is a fatal error.**

Example

```
<?php
function fun1(){
echo "Uday Kumar";
}
fun2();
echo "Fatal Error !!";
?>
```

Output: In the above code we defined a function fun1 but we call another function fun2 i.e. func2 is not defined. So a fatal error will be produced that stops the execution of the script. Like as in the
following image.



4. PARSE ERROR

The parse error occurs if there is a syntax mistake in the script; the output is Parse errors. A parse error stops the execution of the script. There are many reasons for the occurrence of parse errors in PHP. The common reasons for parse errors are as follows:

Common reason of syntax errors are:

a. Unclosed quotes

b. Missing or Extra parentheses

c. Unclosed braces

d. Missing semicolon

Example :

```
<?php
echo "C#";
echo "PHP"
echo "C";
?>
```

Output: In the above code we missed the semicolon in the second line. When that happens there will be a parse or syntax error which stops execution of the script, as in the following image:

PHP Syntax/ Basic structure of PHP

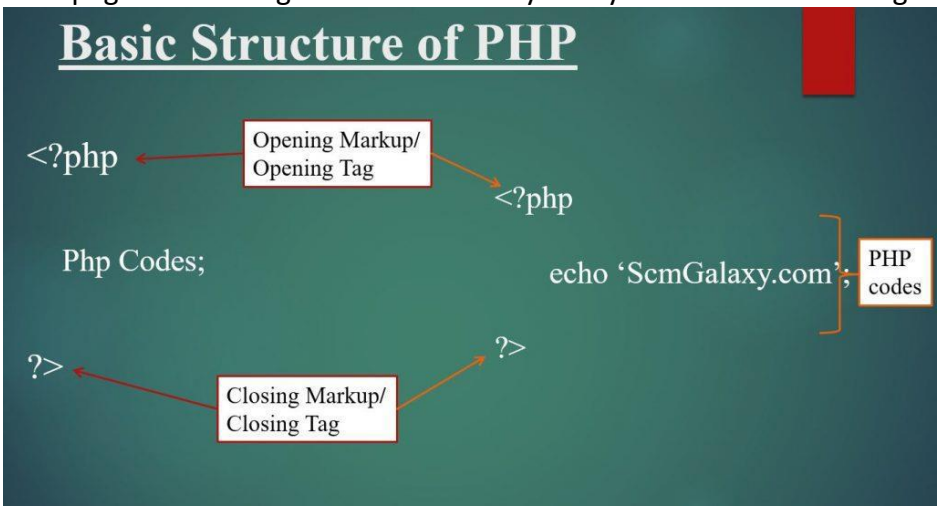
Basic Structure:

PHP code is enclosed within <?php and ?> tags

Statements end with semicolons (;)

Case-sensitive for variables, case-insensitive for keywords
Comments can be single-line (//) or multi-line (/* */)

STRUCTURE OF PHP: PHP is an embedded scripting language when used in Web pages. This means that PHP code is embedded in HTML code. You use HTML tags to enclose the PHP language that you embed in your HTML file — the same way that you would use other HTML tags. We create and edit Web pages containing PHP the same way that you create and edit regular HTML pages.



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 codes go here
?>
Note:

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

HOW TO **RUN PHP** FILE IN XAMPP SOFTWARE PACKAGE

httn://localhost/BCA/nre1.nhn

- 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
- Prgl php is the php file

```
<?php
Echo " hello world";
?>
```

OUTPUT: Hello, Nor1d !";

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, d amicably 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.

2. **print:** Similar to **echo,** the **prin1** statement is used to output data to the browser. It can also output strings and variables.

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.

1. PHP echo

PHP echo is a language construct, not a function. Therefore, you don't need to use parenthesis with it. But if you want to use more than one parameter, it is required to use parenthesis.

The syntax of PHP echo is given below:

1. void echo (string $arg1 [, string $...])

PHP echo statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

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

PHP echo: printing string

```
<?php
echo "Hello by PHP echo";
?>
```

Output:

Hello by PHP echo

PHP echo: printing multi line string

```
<?php
echo "Hello by PHP echo
this is multi line
text printed by
PHP echo statement
";
?>
```

Output:

Hello by PHP echo this is multi line text printed by PHP echo statement

PHP echo: printing escaping characters

```
<?php
echo "Hello escape \"sequence\" characters";
?>
```

Output:

Hello escape "sequence" characters

PHP echo: printing variable value

```
<?php
$msg="Hello JavaTpoint PHP";
echo "Message is: $msg";
?>
```

Output:

Message is: Hello JavaTpoint PHP

2. PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Print statement can be used with or without parentheses: print and print(). Unlike echo, it always returns 1.

The syntax of PHP print is given below:

1. int print(string $arg)

PHP print statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- print is a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than the echo statement.

PHP print: printing string

```
<?php
print "Hello by PHP print ";
print ("Hello by PHP print()");
?>
```

Output:

Hello by PHP print Hello by PHP print()

PHP print: printing multi line string

```
<?php
print "Hello by PHP print
this is multi line
text printed by
PHP print statement
";
?>
```

Output:

Hello by PHP print this is multi line text printed by PHP print statement

PHP print: printing escaping characters

```
<?php
```

```
print "Hello escape \"sequence\" characters by PHP print";
?>
```

Output:

Hello escape "sequence" characters by PHP print

PHP print: printing variable value

```
<?php
$msg="Hello print() in PHP";
print "Message is: $msg";
?>
```

Output:

Message is: Hello print() in PHP

PHP echo and print Statements

We frequently use the echo statement to display the output. There are two basic ways to get the output in PHP:

- echo
- print

echo and print are language constructs, and they never behave like a function. Therefore, there is no requirement for parentheses. However, both the statements can be used with or without parentheses. We can use these statements to output variables or strings.

Difference between echo and print

echo

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

print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

You can see the difference between echo and print statements with the help of the following programs.

For Example (Check multiple arguments)

You can pass multiple arguments separated by a comma (,) in echo. It will not generate any syntax error.

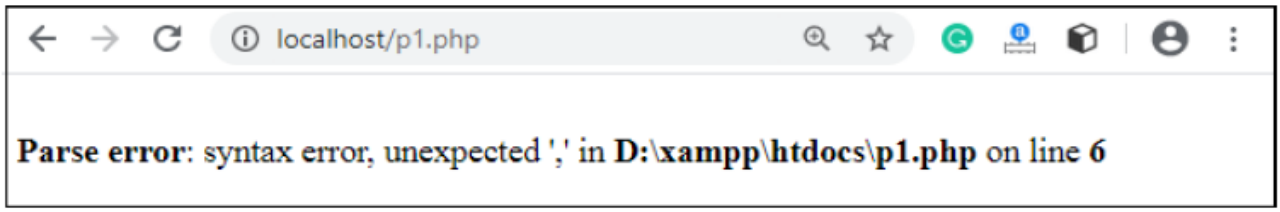
```
<?php
$fname = "Gunjan";
$lname = "Garg";
echo "My name is: ".$fname,$lname;
?>
```

Output:

It will generate a **syntax error** because of multiple arguments in a print statement.

```
<?php
$fname = "Gunjan";
$lname = "Garg";
print "My name is: ".$fname,$lname;
?>
```

Output:

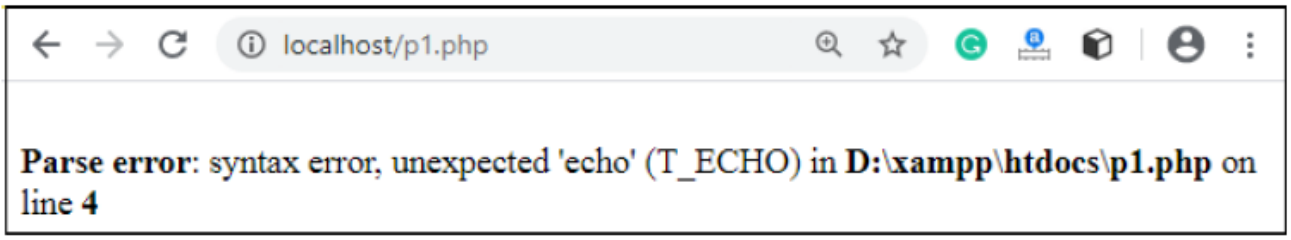


For Example (Check Return Value)

echo statement does not return any value. It will generate an error if you try to display its return value.

```
<?php
$lang = "PHP";
$ret = echo $lang." is a web development language.";
echo "</br>";
echo "Value return by print statement: ".$ret;
?>
```

Output:



As we already discussed that print returns a value, which is always 1.

```
<?php
$lang = "PHP";
$ret = print $lang." is a web development language.";
print "</br>";
```

```
print "Value return by print statement: ".$ret;
?>
```

Output:

3. HTML Integration: Since PHP is often embedded within HTML, you can send HTML code directly to the browser.

```
<?php
$dynamicCOntent = “<p>This is dynamic content generated by PHP.</p>”;
echo “$dynamicContent”;
?>
```

Output:
This is dynamic content generated by PHP.

4. Sending Headers: PHP headers provide additional information to the browser. For example, you can set the content type or redirect the user to another page.

```
<?php
header(“Content-Type: text/plain”);
echo “This is plain text content”;
?>
```

Output:
This is plain text content.

5. Using printf and sprint for formatted output: The printf and sprint functions allow you to format similar to the C language’s printf.

```
<?php
$name=”Jhon”;
$age=25;
printf(“Name: %s, Age: %d”,$name,$age);
?>
```

Output:
Name:Jhon, Age: 25

6. JSON output:

```
<?php
$data=array(“name”=>”Jhon”, “Age”=>25);
header(“Content-Type: application/json”);
echo json_encode($data);
?>
```

Output:
{“name”:Jhon, “Age”: 25}

The data once sent to the browser; it cannot be changed. Headers must be sent before any actual output to the browser, including whitespace.
When working with HTML content, be cautious about security and use functions like html special chars or HTML entities to prevent cross-site scripting (XSS) attacks by escaping special characters.

Data Types Using Php

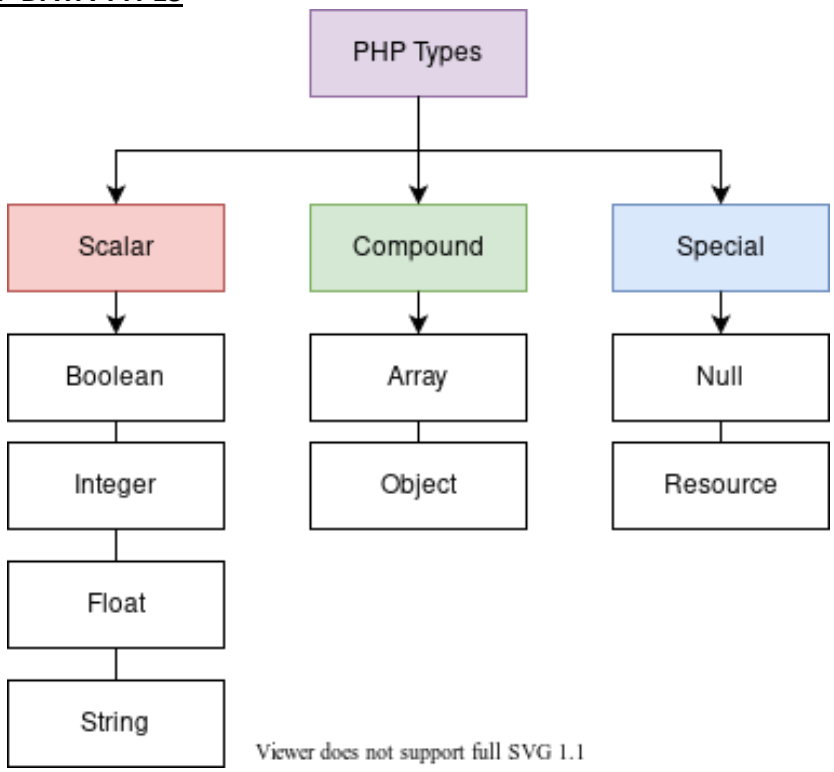
The data type is a fundamental aspect of a programming language, defining what kind of data can be stored and manipulated in a variable or expression.

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 Behavior:** Some PHP functions behave deterrently 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.

PHP DATA TYPES



PHP Data Types

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

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

PHP Data Types: Scalar Types

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

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

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

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

PHP Boolean

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

Example:

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

Output:

This condition is TRUE.

PHP Integer

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

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).

- The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

Example:

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

Output:

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

PHP Float

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

Example:

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

Output:

Addition of floating numbers: 73.812

PHP String

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

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

Example:

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

Output:

Hello Javatpoint
Hello $company

PHP Array

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

Example:

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

Output:

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

PHP object

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

Example:

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

Output:

```
Bike Model: Royal Enfield
```

PHP Resource

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

PHP Null

Null is a special data type that has only one value: **NULL**. There is a convention of writing it in capital letters as it is case sensitive.

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

Example:

```
<?php
$nl = NULL;
echo $nl; //it will not give any output
?>
```

Output:

PHP Keywords

In PH P, keywords are predefined reserved words that have special meanings and can't be used for naming variables, functions, classes. or other identifiers.

Understanding and rising these keywords effectively is essential for writing clean, efficient, and maintainable PH P code.

Some key rules for using keywords in PHP **include:**

1. Reserved **Words:** Keywords arc reserved for specific purposes and cannot be used as identifiers in your code. For example, you cannot create a variable named i I 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.. i I, I F, I r) and they will live the same meaning. However, it's a good practice to use lowercase fear consistency.
3. **Not All Keywords Are Reserved:** Not all words that are part of' PH P's syntax arc 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 their from other elements.

"THE MAIN TYPES OF KE Y WORDS IN PH P:

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 PH P script. They enable developers to make decisions, iteraie Over data, and manage programs how.
2. **Functions and Classes**: Keywords such as function, class, extends, implements, interface, abstract, final, new', and instance of are used to decline functions and classes in PHP. They allow developers to› encapsulate code into reusable modules and create object-oriented structures.
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.
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.
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 rciisc and initialization by allowing developers to split their code into separate files.

6. **Constants:** Keywords like const and define are used to deli ne constants in PH P. 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 iiiiport namespaces in PHP. Namespaces help organize code into logical groupings and prevent naming contacts between different parts of a PHP application.
8. **Variable** Scope: Keywords like global and static are used tt› define variable scope in PH P. They determine the visibility and life time of variables within different parts of a script or application.
9. **Type Declaration:** Keywords like **bool, int, float, string, array, object, resource,** callablc, iterable, and void are used for type declaration in PH P. They allow developers to specify the data type of' variables, parameters, and return values, which can help improve code clarity and maintainability.

PHP has a set of keywords that are reserved words which cannot be used as function names, class names or method names. Prior to PHP 7, these keywords could not be used as class property names either:

Keyword	Description
abstract	Declare a class as abstract
and	A logical operator
as	Used in the foreach loop
break	Break out of loops and switch statements
callable	A data type which can be executed as a function
case	Used in the switch conditional
catch	Used in the try..catch statement
class	Declare a class
clone	Create a copy of an object
const	Define a class constant
continue	Jump to the next iteration of a loop
declare	Set directives for a block of code
default	Used in the switch statement

enddeclare	End a declare block
endfor	End a for block
endforeach	End a foreach block
endif	End an if or elseif block
endswitch	End a switch block
endwhile	End a while block
extends	Extends a class or interface
final	Declare a class, property or method as final
finally	Used in the try...catch statement
fn	Declare an arrow function
for	Create a for loop
foreach	Create a foreach loop
function	Create a function
global	Import variables from the global scope
goto	Jump to a line of code
if	Create a conditional statement
implements	Implement an interface

do	Create a do...while loop
echo	Output text
else	Used in conditional statements
elseif	Used in conditional statements
empty	Check if an expression is empty
return	Exit a function and return a value
static	Declare a property or method as static
switch	Create a switch block
throw	Throw an exception
trait	Declare a trait
try	Create a try...catch structure
unset	Delete a variable or array element
use	Use a namespace
var	Declare a variable
while	Create a while loop or end a do...while loop
xor	A logical operator
yield	Used in generator functions
yield from	Used in generator functions

include	Embed code from another file
include_once	Embed code from another file
instanceof	Test an object's class
insteadof	Resolve conflicts with traits
interface	Declare an interface
isset	Check if a variable exists and is not null
list	Assigns array elements into variables
namespace	Declares a namespace
new	Creates an object
or	A logical operator
print	Output text
private	Declare a property, method or constant as private
protected	Declare a property, method or constant as protected
public	Declare a property, method or constant as public
require	Embed code from another file
require_once	Embed code from another file

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:
1. Using define() function
2. Using const keyword

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

PHP constant: define()

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

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

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

```
<?php
define("MESSAGE","Hello JavaTpoint PHP");
echo MESSAGE;
?>
```

Output:
Hello JavaTpoint PHP

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

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
echo MESSAGE, "</br>";
echo message;
?>
```

Output:
Hello JavaTpoint PHP
Hello JavaTpoint PHP

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

Output:
Hello JavaTpoint PHP
Notice: Use of undefined constant message - assumed 'message'
in C:\wamp\www\vconstant3.php on line 4
message

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

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

```
echo MESSAGE;
?>
```

Output:
Hello const by JavaTpoint PHP

Constant() function
There is another way to print the value of constants using constant() function instead of using the echo statement.
Syntax
The syntax for the following constant function:
1. constant (name)

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

Output:
JavaTpoint
JavaTpoint

Constant vs Variables

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

Magic Constants

Magic constants are the predefined constants in PHP which get changed on the basis of their use. They start with double underscore (__) and ends with double underscore.

They are similar to other predefined constants but as they change their values with the context, they are called **magic** constants.

There are **nine** magic constants in PHP. In which eight magic constants start and end with double underscores (__).

1. __LINE__
2. __FILE__
3. __DIR__
4. __FUNCTION__
5. __CLASS__
6. __TRAIT__
7. __METHOD__
8. __NAMESPACE__
9. ClassName::class

All of the constants are resolved at compile-time instead of run time, unlike the regular constant. Magic constants are case-insensitive

Changelog

Version	Description
5.3.0	Added **__DIR__** and **__NAMESPACE__** magic constant
5.4.0	Added **__TRAIT__** magic constant
5.5.0	Added ::class magic constant

All the constants are defined below with the example code:

1. **__LINE__**

It returns the current line number of the file, where this constant is used.
Example:
Output:

```
Example for __LINE__

You are at line number 4
```

2. ___FILE___:

This magic constant returns the full path of the executed file, where the file is stored. If it is used inside the include, the name of the included file is returned.

Example:

```
<?php
echo "<h3>Example for __FILE__</h3>";
//print full path of file with .php extension
echo __FILE__. "<br><br>";
?>
```

Output:

```
Example for __FILE__

D:\xampp\htdocs\program\magic.php
```

3. ___DIR___:

It returns the full directory path of the executed file. The path returned by this magic constant is equivalent to dirname(__FILE__). This magic constant does not have a trailing slash unless it is a root directory.

Example:

```
<?php
echo "<h3>Example for __DIR__</h3>";
//print full path of directory where script will be placed
echo __DIR__. "<br><br>";
//below output will equivalent to above one.
echo dirname(__FILE__) . "<br><br>";
?>
```

Output:

```
Example for __DIR__

D:\xampp\htdocs\program

D:\xampp\htdocs\program
```

4. ___FUNCTION___:

This magic constant returns the function name, where this constant is used. It will return blank if it is used outside of any function.

Example:

```
<?php
echo "<h3>Example for __FUNCTION__</h3>";
//Using magic constant inside function.
function test(){
//print the function name i.e; test.
echo 'The function name is '. __FUNCTION_____. "<br><br>";
}
test();
9.
//Magic constant used outside function gives the blank output.
function test_function(){
echo 'Hie';
}
test_function();
//give the blank output.
echo  __FUNCTION_____. "<br><br>";
17. ?>
```

Output:

Example for __FUNCTION__

The function name is test

Hie

5. _____CLASS_____:

It returns the class name, where this magic constant is used. _CLASS constant also works in traits.

Example:

```
<?php
echo "<h3>Example for __CLASS__</h3>";
class JTP
{
public function __construct() {
;
}
function getClassName(){
```

```
//print name of the class JTP.
echo __CLASS_____________. "<br><br>";
}
}
$t = new JTP;
$t->getClassName();
//in case of multiple classes
class base
{
function test_first(){
//will always print parent class which is base here.
echo __CLASS__;
22.             }
}
class child extends base
{
public function __construct() {
;
}
}
$t = new child;
$t->test_first();
```

Example for __CLASS___

JTP

?>**Output:**

6. _____TRAIT_:

This magic constant returns the trait name, where it is used.

Example:

```
<?php
echo "<h3>Example for __TRAIT__</h3>";
trait created_trait {
```

```
function jtp(){
//will print name of the trait i.e; created_trait
echo __TRAIT__;
}
}
class Company {
use created_trait;
}
$a = new Company;
$a->jtp();
14. ?>
```

Output:

```
Example for __TRAIT__
created_trait
```

7. _____METHOD_____:

It returns the name of the class method where this magic constant is included. The method name is returned the same as it was declared.

Example:

```
<?php
echo "<h3>Example for __METHOD__</h3>";
class method {
public function __construct() {
//print method::__construct
echo __METHOD_____. "<br><br>";
} public function meth_fun(){
//print method::meth_fun
echo __METHOD__;
}
}
$a = new method;
$a->meth_fun();
?>
```

Output:

```
Example for __METHOD___

method:: construct
method:: meth_fun
```

8. _____NAMESPACE_____:

It returns the current namespace where it is used.

Example:

```
<?php
echo "<h3>Example for __NAMESPACE__</h3>";
class name {
public function __construct() {
echo 'This line will print on calling namespace.';
}
}
$class_name = __NAMESPACE_____. '\name';
$a = new class_name;
?>
```

Output:

```
Example for __NAMESPACE___

This line will print on calling namespace.
```

9. ClassName::class:

This magic constant does not start and end with the double underscore (__). It returns the fully qualified name of the ClassName. ClassName::class is added in **PHP 5.5.0**. It is useful with namespaced classes.

Example:

```
<?php
namespace Technical_Portal;
echo "<h3>Example for CLASSNAME::CLASS </h3>";
class javatpoint {
}
echo javatpoint::class;        //ClassName::class
```

```
?>
```

Output:

```
Example for ClassName::class

Technical  Portal\javatpoint
```

Note: Remember namespace must be the very first statement or after any declare call in the script, otherwise it will generate Fatal error.

PHP Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name. Here, some important points to know about variables:

- 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 of declaring a variable in PHP is given below:

1. $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.

PHP Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

File: variable1.php

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
```

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

Output:

```
string is: hello string
integer is: 200
```

PHP Variable: Sum of two variables

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

Output:

11

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)
```

```
echo "$a <br/> $_b";
?>
```

Output:

```
hello
hello
```

File: variableinvalid.php

```
<?php
$4c="hello";//number (invalid)
$*d="hello";//special
symbol (invalid) 4.
echo "$4c <br/> $*d";
?>
```

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable (T_VARIABLE) or
 '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

PHP Variable Scope

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

- Local variable
- 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.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an

example:

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

Output:

```
Local variable declared inside the function is: 45
```

```
<?php
function mytest()
{
$lang = "PHP";
echo "Web development language: " .$lang;
}
mytest();
//using $lang (local variable) outside the function will generate an error
echo $lang;
?>
```

Output:

```
Web development language: PHP
Notice: Undefined variable: lang in D:\xampp\htdocs\program\p3.php on line 28
```

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.
Let's understand the global variables with the help of an example:

Example:

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

Output:

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

Note: Without using the global keyword, if you try to access a global variable inside the function, it will generate an error that the variable is undefined.

Example:

```
<?php
$name = "Sanaya Sharma";          //global variable
function global_var()
{
echo "Variable inside the function: ". $name;
echo "</br>";
}
global_var();
?>
```

Output:

```
Notice: Undefined variable: name in D:\xampp\htdocs\program\p3.php on line 6
Variable inside the function:
```

Using $GLOBALS instead of global

Another way to use the global variable inside the function is predefined $GLOBALS array.

Example:

```
<?php
        $num1 = 5;          //global variable
        $num2 = 13;         //global variable
        function global_var()
        {
        $sum = $GLOBALS['num1'] + $GLOBALS['num2'];
        echo "Sum of global variables is: " .$sum;
        }
        global_var();
        ?>
```

Output:

Sum of global variables is: 18

If two variables, local and global, have the same name, then the local variable has higher priority than the global variable inside the function.

Example:

```
        <?php
        $x = 5;
        function mytest()
        {
        $x = 7;
        echo "value of x: " .$x;
        }
        mytest();
        ?>
```

Output:

Value of x: 7

Note: local variable has higher priority than the global variable.

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
function static_var()
{
static $num1 = 3;              //static variable
$num2 = 6;                  //Non-static variable
//increment in non-static variable
$num1++;
//increment in static variable
$num2++;
echo "Static: " .$num1 ."</br>";
echo "Non-static: " .$num2 ."</br>";
}
13.
//first function call
static_var();

//second function call
static_var();
?>
```

Output:

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

You have to notice that $num1 regularly increments after each function call, whereas $num2 does not. This is why because $num1 is not a static variable, so it freed its memory after the execution of each function call.

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 arc used to manipulate data, pcrform 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 etrors, and handling exceptional cases gracefully.

TYPES OF EXPRESSION IN PHP

1. Arithmetic Expressions
2. String Expression
3. Comparison 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, //l 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 > Sy), // Comparison expression
echo "Is x greater than y? ” . ($is reater ? "Yes" : "No");
```

4. Logical Expressions: These expressions involve logical operations such as AND (a s), OR (1 |), and NOT (!).

```
Example:
<?php
$age = 25,
$is_adult - ($age >- 18 && $age <= 65); 11
Logical expression echo "Is the person an
adult? " . ($is_adult ? "Yes" : "No");
```

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

```
Example:<?php
$num - IO;
```

```
$cum W- 5; // Assignment expression

echo "Yalue of nuoi qfter assignment: Snum",
```

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 options on arrays, such as accessing elements by index.

Example:

```
<?php
$fruits = ["Apple", "Banana", "Orange”], 11 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 2 + 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 to perform basic arithmetic operations like addition, subtraction, multiplication, division, and modulus.

Operator	*Name*	*Syntax*	*Operation*
+	Addition	$x + $y	Sum the operands
-	Subtraction	$x - $y	Differences in the Operands
*	Multiplication	$x * $y	Product of the operands
/	Division	$x / $y	The quotient of the operands
**	Exponentiation	$x ** $y	$x raised to the power $y
%	Modulus	$x % $y	The remainder of the operands

***Note**: The exponentiation has been introduced in PHP 5.6.*

Example: This example explains the arithmetic operators in PHP.

```
<?php
// Define two numbers
```

```php
$x = 10;
$y = 3;

// Addition
echo "Addition: " . ($x + $y) . "\n";

// Subtraction
echo "Subtraction: " . ($x - $y) . "\n";

// Multiplication
echo "Multiplication: " . ($x * $y) . "\n";

// Division
echo "Division: " . ($x / $y) . "\n";

// Exponentiation
echo "Exponentiation: " . ($x ** $y) . "\n";

// Modulus
echo "Modulus: " . ($x % $y) . "\n";
?>
```

Output

```
Addition: 13
Subtraction: 7
Multiplication: 30
Division: 3.3333333333333
Exponentiation: 1000
Modulus: 1
```

2. Logical Operators

Logical operators are used to operate with conditional statements. These operators evaluate conditions and return a boolean result (true or false).

Operator	Name	Syntax	Operation
and	Logical AND	$x and $y	True if both the operands are true else false
or	Logical OR	$x or $y	True if either of the operands is true otherwise, it is false

Operator	Name	Syntax	Operation
xor	Logical XOR	$x xor $y	True if either of the operands is true and false if both are true
&&	Logical AND	$x && $y	True if both the operands are true else false
||	Logical OR	$x || $y	True if either of the operands is true otherwise, it is false
!	Logical NOT	!$x	True if $x is false

Example: This example describes the logical & relational operators in PHP.

```
<?php
$x = 50;
$y = 30;
 if ($x == 50 and $y == 30)
    echo "and Success \n";

 if ($x == 50 or $y == 20)
    echo "or Success \n";

 if ($x == 50 xor $y == 20)
    echo "xor Success \n";

 if ($x == 50 && $y == 30)
    echo "&& Success \n";

 if ($x == 50 || $y == 20)
    echo "|| Success \n";

 if (!$z)
    echo "! Success \n";
?>
```

Output

```
and Success
or Success
xor Success
&& Success
|| Success
```

! Success

3. Comparison Operators

Comparison operators are used to compare two values and return a Boolean result (true or false).

Operator	Name	Syntax	Operation
==	Equal To	$x == $y	Returns True if both the operands are equal
!=	Not Equal To	$x != $y	Returns True if both the operands are not equal
<>	Not Equal To	$x <> $y	Returns True if both the operands are unequal
===	Identical	$x === $y	Returns True if both the operands are equal and are of the same type
!==	Not Identical	$x == $y	Returns True if both the operands are unequal and are of different types
<	Less Than	$x < $y	Returns True if $x is less than $y
>	Greater Than	$x > $y	Returns True if $x is greater than $y
<=	Less Than or Equal To	$x <= $y	Returns True if $x is less than or equal to $y
>=	Greater Than or Equal To	$x >= $y	Returns True if $x is greater than or equal to $y

Example: This example describes the comparison operator in PHP.

```
<?php
  $a = 80;
  $b = 50;
  $c = "80";

  // Here var_dump function has been used to
  // display structured information. We will learn
  // about this function in complete details in further
  // articles.
  var_dump($a == $c) . "\n";
  var_dump($a != $b) . "\n";
```

```
 var_dump($a <> $b) . "\n";
 var_dump($a === $c) . "\n";
 var_dump($a !== $c) . "\n";
 var_dump($a < $b) . "\n";
 var_dump($a > $b) . "\n";
 var_dump($a <= $b) . "\n";
 var_dump($a >= $b);
?>
```

Output

```
bool(true)

Warning: A non-numeric value encountered in /home/guest/sandbox/Solution.php on line 10
bool(true)

Warning: A non-numeric value encountered in /home/guest/sandbox/Solution.php on line 11
...
```

4. Conditional or Ternary Operators

These operators are used to compare two values and take either of the results simultaneously, depending on whether the outcome is TRUE or FALSE. These are also used as a shorthand notation for the *if...else* statement that we will read in the article on decision making.

Syntax:

```
$var = (condition)? value1 : value2;
```

Here, the condition will either be evaluated as true or false. If the condition evaluates to True, then value1 will be assigned to the variable $var; otherwise, value2 will be assigned to it.

Operator	Name	Operation
?:	Ternary	If the condition is true? then $x : or else $y. This means that if the condition is true, then the left result of the colon is accepted otherwise, the result is on the right.

Example: This example describes the Conditional or Ternary operators in PHP.

```
<?php
 $x = -12;
 echo ($x > 0) ? 'The number is positive' : 'The number is negative';
?>
```

Output

```
The number is negative
```

5. Assignment Operators

Assignment operators are used to assign values to variables. These operators allow you to assign a value and perform operations in a single step.

Operator	Name	Syntax	Operation
=	Assign	$x = $y	Operand on the left obtains the value of the operand on the right
+=	Add then Assign	$x += $y	Simple Addition same as $x = $x + $y
-=	Subtract then Assign	$x -= $y	Simple subtraction same as $x = $x - $y
*=	Multiply then Assign	$x *= $y	Simple product same as $x = $x * $y
/=	Divide, then assign (quotient)	$x /= $y	Simple division same as $x = $x / $y
%=	Divide, then assign (remainder)	$x %= $y	Simple division same as $x = $x % $y

Example: This example describes the assignment operator in PHP.

```
<?php
 // Simple assign operator
 $y = 75;
 echo $y, "\n";

 // Add then assign operator
 $y = 100;
 $y += 200;
 echo $y, "\n";

 // Subtract then assign operator
 $y = 70;
 $y -= 10;
 echo $y, "\n";

 // Multiply then assign operator
 $y = 30;
 $y *= 20;
 echo $y, "\n";

 // Divide then assign(quotient) operator
```

```
 $y = 100;
 $y /= 5;
 echo $y, "\n";

 // Divide then assign(remainder) operator
 $y = 50;
 $y %= 5;
 echo $y;
?>
```

Output

75
300
60
600
20
0

6. Array Operators

These operators are used in the case of arrays. Here are the array operators, along with their syntax and operations, that PHP provides for the array operation.

Operator	Name	Syntax	Operation
+	Union	$x + $y	Union of both, i.e., $x and $y
==	Equality	$x == $y	Returns true if both have the same key-value pair
!=	Inequality	$x != $y	Returns True if both are unequal
===	Identity	$x === $y	Returns True if both have the same key-value pair in the same order and of the same type
!==	Non-Identity	$x !== $y	Returns True if both are not identical to each other
<>	Inequality	$x <> $y	Returns True if both are unequal

Example: This example describes the array operation in PHP.

```
<?php
 $x = array("k" => "Car", "l" => "Bike");
 $y = array("a" => "Train", "b" => "Plane");
```

```
 var_dump($x + $y);
 var_dump($x == $y) . "\n";
 var_dump($x != $y) . "\n";
 var_dump($x <> $y) . "\n";
 var_dump($x === $y) . "\n";
 var_dump($x !== $y) . "\n";
?>
```

Output:

```
array(4) {
 ["k"]=>
 string(3) "Car"
 ["l"]=>
 string(4) "Bike"
 ["a"]=>
 string(5) "Train"
 ["b"]=>
 string(5) "Plane"
}
bool(false)
bool(true)
bool(true)
bool(false)
bool(true)
```

7. Increment/Decrement Operators

These are called the unary operators as they work on single operands. These are used to increment or decrement values.

Operator	Name	Syntax	Operation
++	Pre-Increment	++$x	First, increment $x by one, then return $x
--	Pre-Decrement	--$x	First, decrement $x by one, then return $x
++	Post-Increment	$x++	First returns $x, then increment it by one
--	Post-Decrement	$x--	First, return $x, then decrement it by one

Example: This example describes the Increment/Decrement operators in PHP.

```
<?php
 $x = 2;
 echo ++$x, " First increments then prints \n";
 echo $x, "\n";

 $x = 2;
```

```
 echo $x++, " First prints then increments \n";
 echo $x, "\n";

 $x = 2;
 echo --$x, " First decrements then prints \n";
 echo $x, "\n";

 $x = 2;
 echo $x--, " First prints then decrements \n";
 echo $x;
?>
```

Output

```
3 First increments then prints
3
2 First prints then increments
3
1 First decrements then prints
1
2 First prints then decrements
1
```

8. String Operators

This operator is used for the concatenation of 2 or more strings using the concatenation operator ('.'). We can also use the concatenating assignment operator ('.=') to append the argument on the right side to the argument on the left side.

Operator	Name	Syntax	Operation
.	Concatenation	$x.$y	Concatenated $x and $y
.=	Concatenation and assignment	$x.=$y	First, it concatenates then assigns, the same as $x = $x.$y

Example: This example describes the string operator in PHP.

```
<?php
 $x = "Geeks";
 $y = "for";
 $z = "Geeks!!!";
 echo $x . $y . $z, "\n";
 $x .= $y . $z;
 echo $x;
?>
```

Output

GeeksforGeeks!!!
GeeksforGeeks!!!

9. Type Operators:

Type operators are used to check the type of a variable. The type operator instance of 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 = ’1s -1’;
```

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
$i cent t =  g?!fi le  get  contents('nonexistent_file.txt');
// Set error i'eportinÿ level
crror  reporting(E  ALL &     E  NOTICE):
```

PHP OPERATORS PRECEDENCE

Operator Precedence:

Operator precedence determines which operation is performed first in an expression with more than one operator with different precedence. Operators with higher precedence are evaluated before those with lower precedence.

Operator Associativity:

Operator associativity defines the direction in which operators of the same precedence are evaluated when they appear in an expression. It can be either left-to-right or right-to-left.

Operator Precedence and Associativity Table:

The following tables list the operator precedence from highest to lowest and the associativity for each of the operators:

Precedence	Operator	Description	Associativity
1	()	Parentheses (function call)	Left-to-Right
	[]	Array Subscript (Square Brackets)	
	.	Dot Operator	
	->	Structure Pointer Operator	
	++ , —	Postfix increment, decrement	
2	++ / —	Prefix increment, decrement	Right-to-Left
	+ / –	Unary plus, minus	
	! , ~	Logical NOT, Bitwise complement	
	(type)	Cast Operator	
	*	Dereference Operator	

Precedence	Operator	Description	Associativity
	&	Addressof Operator	
	sizeof	Determine size in bytes	
3	*,/,%	Multiplication, division, modulus	Left-to-Right
4	+/-	Addition, subtraction	Left-to-Right
5	<< , >>	Bitwise shift left, Bitwise shift right	Left-to-Right
6	< , <=	Relational less than, less than or equal to	Left-to-Right
	> , >=	Relational greater than, greater than or equal to	
7	== , !=	Relational is equal to, is not equal to	Left-to-Right
8	&	Bitwise AND	Left-to-Right
9	^	Bitwise exclusive OR	Left-to-Right
10	|	Bitwise inclusive OR	Left-to-Right
11	&&	Logical AND	Left-to-Right
12	||	Logical OR	Left-to-Right
13	?:	Ternary conditional	Right-to-Left
14	=	Assignment	Right-to-Left
	+= , -=	Addition, subtraction assignment	
	*= , /=	Multiplication, division assignment	
	%= , &=	Modulus, bitwise AND assignment	

Precedence	Operator	Description	Associativity
	^= , |=	Bitwise exclusive, inclusive OR assignment	
	<<=, >>=	Bitwise shift left, right assignment	
15	,	comma (expression separator)	Left-to-Right

Difference between Operator Precedence and Operator Associativity:

Aspect	Operator Precedence	Operator Associativity
Definition	Determines the order of evaluation of operators in an expression.	Defines the direction of evaluation when operators of the same precedence appear.
Priority	Higher precedence operators are evaluated before lower precedence operators.	Associativity determines the order in which operators are evaluated within an expression of the same precedence.
Example	In 2 + 3 * 4, * has higher precedence than +, so 3 * 4 is evaluated first.	In 2 + 3 + 4, left-associative operators are evaluated from left to right.
Representation	Precedence levels are often specified in language documentation or expressed with parentheses in expressions.	Associativity is usually implicit and is determined by the language specification.