NEW SYLLABUS

PG TRB COMPUTER SCIENCE

PHP AND MYSQL

PG TRB

COMPUTER SCIENCE

UNIT - IX

PHP AND MYSQL

Professor Academy

Title : PG TRB COMPUTER SCIENCE
Unit IX - PHP AND MYSQL

Edition: 1" edition
Year : 2025

Published by:

Professor Academy
No:14, West Avenue, Taylor Estate,
Kodambakkam, Chennai-600 024.
7070701005, 7070701009
professoracademy.com

Feel free to mail us at **feedback@professoracademy.com**

INDEX

SYLLABUS

Basics of PHP:

Evaluation of PHP, Basic Syntax, Defining variable and constant, PHP Data type, Operator and Expression, Making Decisions, Doing Repetitive task with looping, Mixing Decisions and looping with HTML.

Functions:

Defining a function, Call by value and Call by reference, Recursive function.

String Handling:

Creating and accessing, String Searching and Replacing String, Formatting String, String Related Library functions.

Array:

Anatomy of an Array, Index based and Associative array, Accessing array, Element Looping with Index based array, Looping with associative array.

Working with file and Directories:

Understanding file and directory, Opening and closing a file, Copying, renaming and deleting a file, working with directories, Creating and deleting folder, File uploading and downloading.

MySQL:

MySQL database connection, Creating a table with key constraints, dropping a table, adding, retrieving, updating data, deleting data, Performing additional queries (Joins and subqueries), Connecting to MySQL, Accessing MySQL using PHP, Querying MySQL database with PHP.

Professor Academy

PG TRB

COMPUTER SCIENCE

COURSE DETAILS

+91 707070 1005
+91 707070 1009

www.professoracademy.com

Professor Academy

PG TRB 2025 ONLINE COURSE

New Syllabus 2024

Complete coverage of **New Syllabus 2024**

Online Live Classes

Online Live Classes
200+ hrs. of Lectures

100+ Test Series

- Daily test
- Unit wise test
- Full-length test

REC

Recorded Access

Recorded Access
24/7 Availability

Study Material

Study Material
12 Printed Books + Class notes

Support

Technical & Academic Support

www.professoracademy.com

Introduction to PHP

Introduction

The term PHP is an acronym for – Hypertext Preprocessor. PHP is a server-side scripting language designed specifically for web development. It is an open-source which means it is free to download and use. It is very simple to learn and use. The file extension of PHP is ".php".

What is PHP?

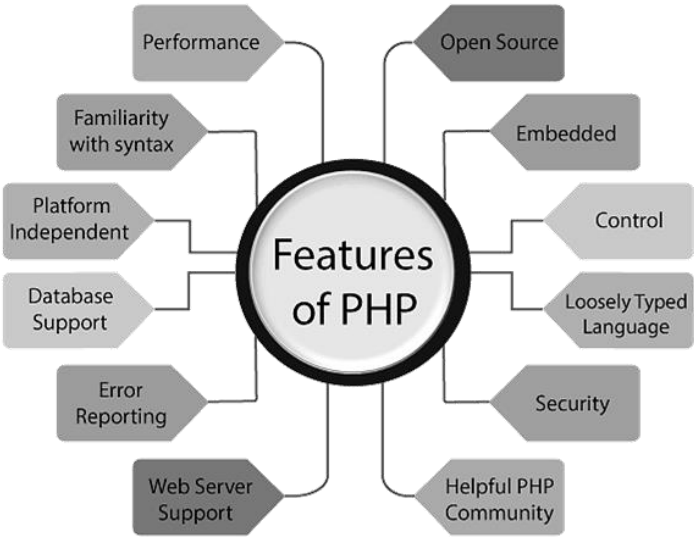
PHP is a server-side scripting language created primarily for web development but it is also used as a general-purpose programming language. Unlike client-side languages like JavaScript, which are executed on the user's browser, PHP scripts run on the server. The results are then sent to the client's web browser as plain HTML.

History of PHP

PHP was introduced by Rasmus Lerdorf in 1994, the first version and participated in the later versions. It is an interpreted language and it does not require a compiler. The language quickly evolved and was given the name "PHP," which initially named was "Personal Home Page."

- PHP 3 (1998): The first version considered suitable for widespread use.
- PHP 4 (2000): Improved performance and the introduction of the Zend Engine.
- PHP 5 (2004): Added object-oriented programming features.
- PHP 7 (2015): Significant performance improvements and reduced memory usage.
- PHP 8 (2020): Introduction of Just-In-Time (JIT) compilation, further enhancing performance.

PHP Features



Performance:

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source:

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax:

PHP has easily understandable syntax. Programmers are comfortable coding with it.

Embedded:

PHP code can be easily embedded within HTML tags and script.

Platform Independent:

PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

Database Support:

PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Error Reporting -

PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Loosely Typed Language:

PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Web servers Support:

PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

Security:

PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

Control:

Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

A Helpful PHP Community:

It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

Interesting fact: PHP Was Created by a Programmer Who Didn't Know What He Was Doing

- **Rasmus Lerdorf**, the creator of PHP, originally wrote it in 1993 as a simple set of scripts to track visitors to his online resume. The first version was called **"Personal Home Page Tools"**, which later evolved into **PHP/FI** (Personal Home Page/Form Interpreter).
- He didn't intend for it to become a full-fledged programming language. In fact, PHP was initially just a collection of CGI (Common Gateway Interface) binaries for tracking web visitors.

How PHP Works?

PHP scripts are executed on the server. Here's a typical flow of how PHP works:

- A user requests a PHP page via their web browser.
- The server processes the PHP code. The PHP interpreter parses the script, executes the code, and generates HTML output.
- The server sends the generated HTML back to the client's browser, which renders the web page.

This server-side processing allows for dynamic content generation and ensures that sensitive code is not exposed to the client.

Syntax

```
<?php
  // PHP code goes here
?>
```

Basic Example of PHP

HTML

```
<!DOCTYPE html>
<html>
  <head>
   <title>PHP Hello World</title>
</head>
<body>
   <?php echo "Hello, World! This is PHP code";?>
</body>
</html>
```

Output:

Hello, World! This is PHP code

PHP Installation and Configuration

To get started with PHP, you need to set up a local environment. Here's a simple guide:

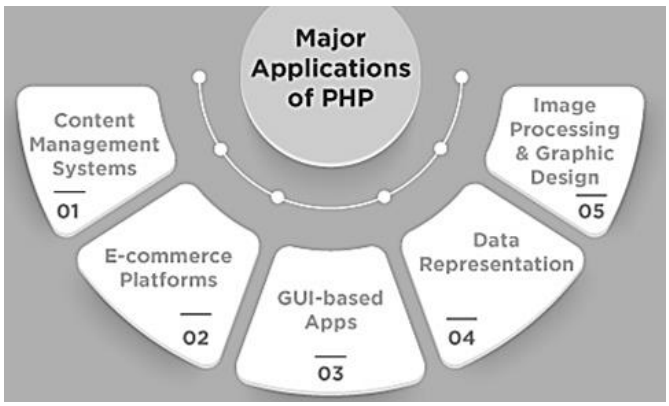
- **Install a Web Server:** Apache or Nginx are popular choices.
- **Install PHP:** Download and install the latest version of PHP from the official PHP website.
- **Install a Database:** MySQL or MariaDB are commonly used with PHP.
- **Configure PHP:** Update the php.ini file to configure PHP settings as needed.

For beginners, a complete package like XAMPP (Windows) or MAMP (macOS) is recommended, as it comes bundled with Apache, PHP, and MySQL.

Interesting fact: PHP Is Everywhere

- Over 75% of websites on the internet use PHP in some capacity. This includes some of the most popular websites in the world like Facebook, WordPress, Wikipedia, and Tumblr.
- It's used for creating dynamic web pages, handling forms, managing databases, and more.

Applications of PHP



PHP is versatile and can be used in a variety of web development scenarios, including:

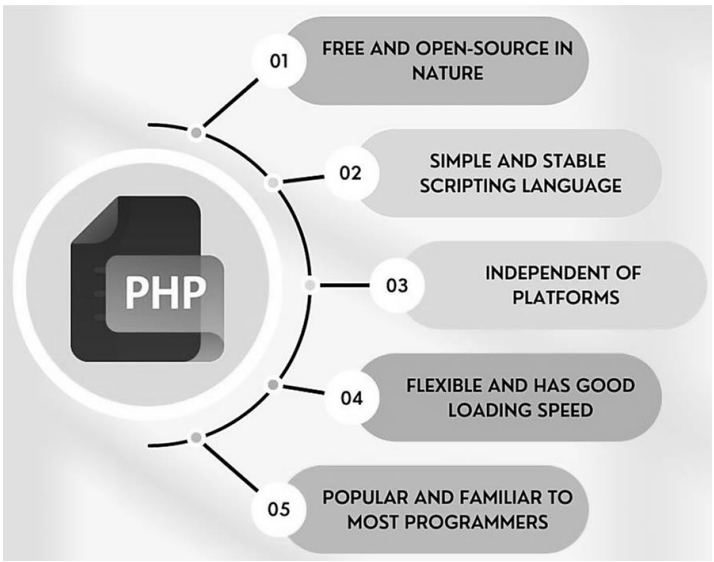
- **Content Management Systems (CMS):** Many popular CMSs like WordPress, Joomla, and Drupal are built with PHP.
- **E-commerce Platforms**: PHP is commonly used to develop e-commerce websites due to its database integration capabilities.
- **GUI-based Apps**: PHP is also used to create graphical user interface-based applications for desktops. Several tools, such as PHP-GTK 2 and ZZEE PHP GUI, are used for scripting client-side GUI-based apps.
- **Data Representation**: The developers often use PHP for data representation purposes. Charts, scatter-dot plots, graphs, etc.,
- **Image Processing & Graphics Design:** popularly used in the processing of graphics and images. Certain image processing libraries such as Imagine, ImageMagick, and GD library can be integrated with PHP for various purposes ranging from resizing and rotating to cropping and thumbnail creation.

Why should we use PHP?

PHP can actually do anything related to server-side scripting or more popularly known as the backend of a website. For example, PHP can receive data from forms, generate dynamic page content, can work with databases, create sessions, send and receive cookies, send emails, etc. There are also many hash functions available in PHP to encrypt users' data which makes PHP secure and reliable to be used as a server-side scripting language. So these are some of PHP's abilities that make it suitable to be used as a server-side scripting language.

Even if the above abilities do not convince you of PHP, there are some more features of PHP. PHP can run on all major operating systems like Windows, Linux, Unix, Mac OS X, etc. Almost all of the major servers available today like Apache supports PHP. PHP allows using a wide range of databases. And the most important factor is that it is free to use and download and anyone can download PHP from its

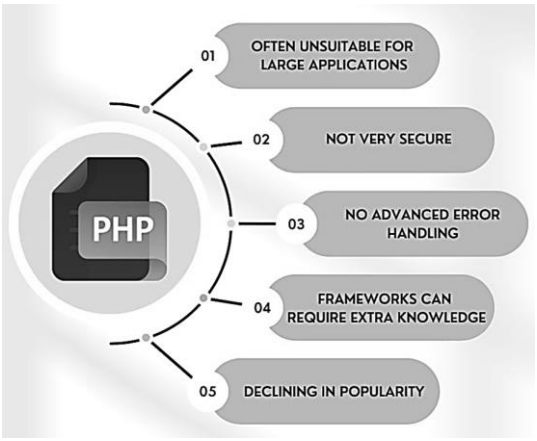
Advantages of PHP



- **Open Source**: PHP is an open-source language, meaning it is freely available for anyone to use and distribute. This openness has fostered a large and active community of developers who continuously contribute to its growth, improvement, and feature development.
- **Easy to Learn**: The syntax of PHP is quite similar to C and other programming languages. This makes PHP relatively easy to learn, especially for developers who already have some programming experience. Beginners find it approachable due to its straightforward syntax and extensive online resources.

- **Web Integration**: PHP is designed specifically for web development and is embedded within HTML. It seamlessly integrates with various web technologies, facilitating the creation of dynamic and interactive web pages.
- **Database Support**: PHP has excellent support for various databases, including MySQL, PostgreSQL, SQLite, and more. This makes it easy to connect and interact with databases, a crucial aspect of many web applications.
- **Cross-Platform Compatibility**: PHP is platform-independent and runs on various operating systems, including Windows, Linux, macOS, and others. This ensures compatibility across different environments.
- **Large Community and Documentation**: PHP has a vast and active community of developers. The abundance of online resources, tutorials, and documentation makes it easier for developers to find solutions and seek help when needed.
- **Frameworks and CMS**: There are popular PHP frameworks like Laravel, Symfony, and CodeIgniter, which provide pre-built modules and features, aiding in rapid development. Additionally, PHP supports widely used content management systems (CMS) like WordPress and Joomla.
- **Server-Side Scripting**: PHP scripts are executed on the server, reducing the load on the client's side. This server-side scripting capability is crucial for generating dynamic content and performing server-related tasks.
- **Community Support**: The PHP community actively contributes to the language's development, ensuring regular updates, security patches, and improvements.

Disadvantages of PHP



- **Inconsistency:** PHP has been criticized for inconsistencies in function names and parameter orders. This can lead to confusion for developers, especially when working with a mix of older and newer functions.
- **Security Concerns**: If not handled properly, PHP code may be susceptible to security vulnerabilities, such as SQL injection and cross-site scripting (XSS). Developers need to be cautious and follow best practices to secure PHP applications.
- **Performance**: While PHP performs well for many web applications, it may not be as fast as some compiled languages like C or Java. However, advancements and optimizations in recent versions have improved performance.
- **Lack of Modern Features**: Compared to newer languages, PHP may lack some modern language features. However, recent versions of PHP have introduced improvements and features to address this concern.
- **Scalability Challenges**: PHP can face challenges when it comes to scaling large and complex applications. Developers may need to adopt additional tools or frameworks to address scalability issues.

Success is not for the ones who quit, but for those who persist." 💪📚

- **Not Suitable for Large-Scale Applications**: While PHP is suitable for small to medium-sized projects, it might not be the best choice for extremely large and complex applications where more structured languages might be preferred.
- **Limited Object-Oriented Programming (OOP) Support**: Although PHP supports OOP, its implementation has been criticized for not being as robust as in some other languages. However, recent versions have introduced improvements to enhance OOP capabilities.

Evaluation of PHP

HP or Hypertext Preprocessor was developed by an employed Danish programmer, Rasmus Lerdorf out of Toronto in Canada in 1994 and released to the public in 1995 with the name Personal Home Page tools. It was then rewritten in 1996 by Zeev Suraski and Andi Gutman who launched it as PHP3. In 2000, PHP4 was released incorporated into a scripting engine followed by 3 more major-version and few subversions until the latest version PHP7.0 was released in 2015. It gained popularity over the years owing to it being open source and allowing other programmers to use the language on their own pages and today controls over 80% of websites across the globe.

PHP has emerged as one of the highly popular server-side scripting languages owing to its flexibility, innovative features and it is the fastest resource present for creating database-enabled dynamic websites. PHP includes a well-organized code that is easily embedded into HTML code and all features and updates to the language are available free of cost. In comparison to the other languages, troubleshooting and debugging issues is easier in PHP. Moreover, it supports major operating systems like Windows, Linux, Unix, Mac OS, etc and also supports enterprise and web servers like Microsoft IIS, Netscape, Apache, etc.

These features of PHP made it increasingly the choice for web developers when designing complex but attractive websites during a short span of time.

Better run-time performance and improved extension API with web server abstraction layers and the methodology of compile first and execute later were some of the reasons which made PHP the base for building CMS like WordPress in 2003. Over time, with each update, improvements in error handling, performance improvements almost twice as the previous versions and several more features led to a continuous increase in the popularity of the language as the preferred web development technology.

Basic Syntax of PHP

PHP, a powerful server-side scripting language used in web development. It's simplicity and ease of use makes it an ideal choice for beginners and experienced developers. This article provides an overview of PHP syntax. PHP scripts can be written anywhere in the document within PHP tags along with normal HTML.

PHP code is executed between PHP tags, allowing the integration of PHP code within HTML. The most common PHP tag is <?php … ?>, which is used to enclose PHP code. The <?php ….?>is called Escaping to PHP.

```
<?php
    // code
?>
```

The script starts with <?php and ends with ?>. These tags are also called 'Canonical PHP tags'. Everything outside of a pair of opening and closing tags is ignored by the PHP parser. The open and closing tags are called delimiters. Every PHP command ends with a semi-colon (;).

Basic Example of PHP

```
1  <?php
2
3  // Here echo command is used to print
4  echo "Hello, world!";
5
6  ?>
```

Output
Hello, world!

Embedding PHP in HTML

PHP code can be embedded within HTML using the standard PHP tags. In this example, the <?php echo “Hello, PHP!”; ?> statement dynamically inserts a heading into the HTML document.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PHP Syntax Example</title>
</head>
<body>
    <h1><?php echo "Hello, PHP!"; ?></h1>
</body>
</html>
```

Did You Know PHP is Case-insensitive for Functions?

- PHP function names are **case-insensitive**. Whether you write echo, ECHO, or Echo, the result will be the same. However, it's best practice to follow standard conventions and use lowercase for function names.

SGML or Short HTML Tags

These are the shortest option to initialize a PHP code. The script starts with <? and ends with ?>. This will only work by setting the *short_open_tag* setting in the *php.ini* file to ‘on’.

Example:

```
<?php
// Here echo command will only work if
// setting is done as said before
echo "Hello, world!";
?>
```

Output

Hello, world!

Case Sensitivity

PHP is partially case-sensitive-

- Keywords (like if, else, while, echo) are not case-sensitive.
- Variable names are case-sensitive.

Example:

```
<?php
$Var = "Hello Teacher";
// Outputs: Hello Teacher
echo $Var;
// Error: Undefined variable $var
echo $var;
?>
```

Interesting fact: PHP is Not Just for Websites

While PHP is mostly associated with web development, it can be used for other tasks as well. PHP has been used for creating **command-line scripts**, running **background tasks**, or even building **desktop applications** (though it's not commonly used for that purpose).

Comments in PHP

Comments are used to make code more readable by explaining the purpose of specific code blocks. Comments are ignored by the PHP interpreter.

Single Line Comment

As the name suggests, these are single line or short relevant explanations that one can add to their code. To add this, we need to begin the line with (//) or (#).

```
<?php
// This is a single line comment
// These cannot be extended to more lines
echo "Hello World!";
# This is also a single line comment
?>
```

Output

Hello World!

Multi-Line or Multiple Line Comment

It is used to accommodate multiple lines with a single tag and can be extended to many lines as required by the user. To add this, we need to begin and end the line with (/*…*/)

```
<?php

/* This is a multi line comment
   In PHP variables are written
   by adding a $ sign at the
   beginning.*/

$geek = "Hello World!";
echo $geek;

?>
```

Output

Hello World!

Variables and Data Types

Variables are used to store data that can be manipulated within PHP. They are declared using the $ symbol followed by the variable name.

Declaring Variables

Variables are created by assigning a value to them using the assignment operator (=).

PHP

```php
$name = "XYZ";  // String
$age = 30;      // Integer
$isEmployed = true; // Boolean
```

Data Types

PHP supports several data types, including:

- **String:** A sequence of characters.
- **Integer:** Whole numbers.
- **Float (Double):** Numbers with a decimal point.
- **Boolean**: Represents true or false.
- **Array**: A collection of values.
- **Object:** An instance of a class.
- **NULL**: A special type representing a variable with no value.
- **Resource**: A special type that holds a reference to external resources (like database connections).

Blocks in PHP

In PHP, multiple statements can be executed simultaneously (under a single condition or loop) by using curly-braces ({}). This forms a block of statements that gets executed simultaneously.

```
<?php
$var = 50;
if ($var > 0) {
   echo ("Positive as \n");
   echo ("greater than 0");
}
?>
```

Output

Positive as
greater than 0

Virtual Resources

Scan the QR code to unlock extra content

Practice Questions

1. What does PHP stand for?
A) Personal Home Page
B) PHP Hypertext Preprocessor
C) Private Home Page
D) Public Hypertext Processor

2. Which of the following is a correct way to start a PHP code block?
A) <php> B) <?php
C) <script> D) <html>

3. Which function is used to output data to the browser in PHP?
A) echo() B) print()
C) output() D) Both A) and B)

4. What is the default file extension for PHP files?
A) .html B) php C) txt D) xml

5. Which of the following is NOT a valid variable name in PHP?
A) $myVariable B) $_myVar
C) $1stVariable D) $my_var

6. What is the correct way to create a constant in PHP?
A) define("CONSTANT_NAME", "value");
B) const CONSTANT_NAME = "value";
C) Both A) and B)
D) None of the above

7. Which of the following statements is true about PHP?
A) PHP is a server-side scripting language.
B) PHP can be embedded in HTML.
C) PHP is platform-independent.
D) All of the above.

8. What is the purpose of the PHP require() function?
A) To include a file only once.
B) To include a file and terminate the script if it fails.
C) To include a file and continue execution if it fails.
D) To create a new PHP variable.

9. Which PHP super global is used to collect form data after submitting an HTML form?
A) $_POST B) $_GET
C) $_REQUEST D) All of the above

10. In PHP, how do you create an associative array?
A) $arr = array("key1" => "value1", "key2" => "value2");
B) $arr = ["key1" => "value1", "key2" => "value2"];
C) Both A) and B)
D) None of the above

11. What is one of the primary advantages of using PHP for web development?
A) It requires extensive configuration
B) It is highly compatible with various databases
C) It has limited community support
D) It only runs on Windows servers

12. Which of the following is a disadvantage of PHP?
A) It is completely free to use
B) It can lead to security vulnerabilities if not coded properly
C) It supports only object-oriented programming
D) It has a very steep learning curve

13. In which scenario is PHP commonly applied?
A) Desktop application development
B) Creating static websites without interactivity
C) Server-side scripting for dynamic web applications
D) Developing mobile applications exclusively

14. One of the reasons PHP is popular among developers is its:
A) Lack of frameworks
B) Rich set of built-in functions
C) Requirement for high-end servers
D) Incompatibility with HTML

15. What is a typical application of PHP in a content management system (CMS)?
A) Handling client-side animations
B) Managing database interactions for content storage
C) Writing operating system-level scripts
D) Developing hardware drivers

Ans : 1-b,2-b,3-d,4-b,5-c,6-c,7-d,8-b,9-d,10-c,11-b,12-b,13-c,14-b,15-b

Professor Academy

VIRTUAL CLASS FEATURES

Start Date: December 06, 2024

Duration of Classes: 5 months (up to exam date)

Class Schedule:

- **Morning Session:** 5:00 AM - 7:00 AM
- **Evening Session:** 7:30 PM - 9:00 PM

(subject to change based on exam announcements)

Key Features:

- Interactive online live classes
- Weekly schedule updates via WhatsApp community
- Sessions conducted through Zoom for ease of access

Professor Academy

COURSE BENEFITS

APP FEATURES

- Login ID and password will be provided to the candidates to access the mobile app after enrolling in course.
- **Access to Missed Live Sessions** as Recorded Videos.
- Validity for Recorded Sessions and Test Series will be provided upto 1 year in our mobile app (Upto date of the examination)
- **Access to Missed Live Sessions as Recorded Videos.**
- **Mobile App** for learning (For Android users)
- **Website** for easy access from any device

2. Variables and Constants in PHP

Defining Variables

A PHP variable is a name given to a memory address that holds data. The basic method to declare a PHP variable is by using a $ sign which is followed by the variable name. A variable helps the PHP code to store information in the middle of the program. If we use a variable before it is assigned then it already has a default value stored in it. Some of the data types that we can use to construct a variable are integers, doubles, and boolean.

Example:

```
<?php
$txt = "Hello";
$x = 5;
$y = 10.5;
echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
```

Output

Hello
5
10.5

Rules for variable names:

- Variable names in PHP start with a dollar ($) sign followed by the variable name.
- Variable name can contain alphanumeric characters and underscore (_).
- Variable names must start with a letter or an underscore (_). (For eg: $abc, $x1, $_g, $abc_1 etc.)
- Variable names cannot start with a number.
- Variable names are case-sensitive. (for eg: $x and $X are treated as two different variables.)
- In PHP we don't use any command to declare variables.
- A variable is created as soon as you assign a value to it.
- A variable takes a datatype according to the value assigned to it.
- Since we don't have to specify datatypes for PHP variables, PHP is called as loosely typed language.

Exam Points PHP Variables Don't Need Type Declarations:

- PHP is a **loosely typed language**, which means you can change a variable's type at any point. For example, a variable that initially holds an integer can later hold a string or an array.

Scope of variables:

Variables can be declared anywhere in the program.

Scope of a variable is a part of the program where the variable is accessible.

PHP has three different variable scopes:

- Local
- Global
- Static

Local scope:

A variable declared within a function has a local scope and can be accessed within a function only.A function is a small program performing a particular task which is called when required.

Global scope:

A variable declared outside a function has a global scope and can be accessed outside the function only.Actually, global variables can be accessed anywhere using the global keyword.

Static scope:

A variable declared with static keyword is said to have static scope within the function.
Normally when variables are executed, they lose their values or memory.

But when a variable is declared as static, it doesn't lose its value. It remains static within multiple function calls.

(Content Developed by Pro.fessor Ac.ademy)

Variable declaration:

Variable is declared as follows:

```
$variable_name=value;
```

Example of variable declaration is given below:

```
$x=5;
```

$x is a variable and 5 is a value assigned to $x variable using assignment operator (=). The assignment operator assigns the right hand side value to the left hand side variable in an expression.

The variable name can be just alphabets or they can be some descriptive names like $school_name, $names, $games etc.

In PHP we can print a value of variable using an echo statement as follows:

```
<?php
$x=10;
echo $x;
?>
```

Output

10

Demonstration of Global Scope of variables:

Write the following code in index.php file and test it by putting it in the newly created folder var_constant in htdocs folder of xampp folder.

The code is as follows:

```
<?php
$x = 10; // Global variable
function myfun()
{
   global $x; // Declare $x as global
   $y = 20; // Local variable
   echo "Value of var x inside myfun = " . $x;
   echo "<br>";
   echo "Value of var y inside myfun = " . $y;
   echo "<br>";
}
myfun();
echo "Value of var x outside myfun = " . $x;
echo "<br>";
?>
```

Here, we have declared a global variable and a local variable in the function myfun().

- Both the value of $x and $y are printed in the function myfun() as well as outside the function.
- The function myfun() needs to be called for execution as done in the statement myfun().

```
Value of var x inside myfun =
Value of var y inside myfun = 20
Value of var x outside myfun = 10
Value of var y outside myfun =
```

- You will get the following error shown in the figure below:
- This error occurs because the global variable $x is not accessible in the function myfun().
- And local variable $y of function myfun() is not accessible outside the function.
- Now just comment the following statements given below in the code:
- echo "Value of var x inside myfun = ".$x;
- echo "Value of var y outside myfun = ".$y;
    - We will get the following output:

```
Value of var x inside myfun = 10
Value of var y inside myfun = 20
Value of var x outside myfun = 10
```

- But meaning of global is accessible everywhere, and here we see that the global variable $x is not accessible inside the function myfun().
- We can make it accessible by using keyword global before the variable $x inside function myfun().

It is shown below:

```
<?php
$x=10;  //global variable
//echo $x;
function myfun()
{
  global $x;  //accessing global variable inside function
  $y=20;  //local variable

  echo "Value of var x inside myfun = ".$x;
  echo "<br>";
  echo "Value of var y inside myfun = ".$y;
  echo "<br>";
}
myfun();
echo "Value of var x outside myfun = ".$x;
```

```
echo "<br>";
//echo "Value of var y outside myfun = ".$y;
?>
```

- In the above code we have a statement global $x; written inside the function myfun(). This allows access to global variable inside the function.
- Now let us see the output of the above code:

```
Value of var x inside myfun = 10
Value of var y inside myfun = 20
Value of var x outside myfun = 10
```

- The statement Value of var x inside myfun = 10 proves that now the value of $x is accessible in function myfun().
- The statement
- echo "Value of var y outside myfun = ".$y;
- is commented in the program because it will give error since the local variables are not accessible outside the function.
- You might have noticed a period (.) in the echo statement. For example let us see the following statement:
- echo "Value of var y outside myfun = ".$y;
- Here we have a period (.) in between a string "Value of var y outside myfun = " and variable $y. This period is used for concatenating/joining two values.
- Demonstration of Static Scope of variables:
- We discussed that static scope means the value of a variable is retained within multiple function calls.
- Let us try to demonstrate it.
- Write the following code in index.php file by commenting all the previous code:

```
<?php
function static_eg()
{
        $x=0;
        static $y=0;
echo "non-static var x = $x    static var y = $y <br>";
        $x++;
        $y++;
}
static_eg();
static_eg();
static_eg();
static_eg();
?>
```

- Here, we see that we have a function static_eg() that contains two variables viz. $x and $y.
- $x is a simple local variable and $y is a local variable but also a static variable.
- Both are initialized to zero.
- After printing the values of both the variables using echo statement, they are incremented each time.
- The function static_eg() is called 4 times.
- The output of the above code is given below:

```
non-static var x = 0    static var y = 0
non-static var x = 0    static var y = 1
non-static var x = 0    static var y = 2
non-static var x = 0    static var y = 3
```

- Here, we can see that the value of variable $x is zero (0) every time and the value of variable $y is incremented by 1 each time.

Interesting Facts: You Can Use unset() to Destroy Variables:

- PHP allows you to **destroy variables** with the unset() function. After calling unset(), the variable is no longer available.

php

Copy code

```
$x = 10;

unset($x);  // $x is now destroyed
```

- This is because a simple variable loses its value once it comes out of the block it is defined in, but a static variable retains its value each time.

Defining Constants

PHP Constants are the identifiers that remain the same. Usually, it does not change during the execution of the script. They are case-sensitive. By default, constant identifiers are always uppercase. Usually, a Constant name starts with an underscore or a letter which is followed by a number of letters and numbers. They are no need to write a constant with the $ sign. The constant() function is used to return the value of a constant.

Example

```
<?php
define("Hello", "Welcome ");
echo Hello;
?>
```

Output

Welcome

Difference between PHP Constants and PHP Variables

PHP Constants	PHP Variables
In PHP constants there is no need to use $ sign.	In PHP Variables the $ sign is been used.
The data type of PHP constant cannot be changed during the execution of the script.	The data type of the PHP variable can be changed during the execution of the script.
A PHP constant once defined cannot be redefined.	A PHP variable can be undefined as well as can be redefined.
We can not define a constant using any simple assignment operator rather it can only be defined using define().	We can define a variable using a simple assignment operation(=).
Usually, constants are written in numbers.	On the other hand, variables are written in letters and symbols.
PHP constants are automatically global across the entire script.	PHP variables are not automatically global in the entire script.
PHP constant is comparatively slower than PHP variable	A PHP variable is comparatively faster than the PHP constant

Exam Points Constants Can't Be Changed Once Defined:

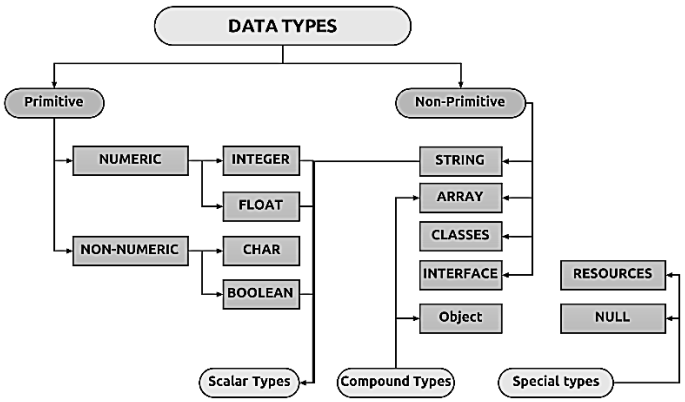
- Once a constant is defined using define(), it **cannot** be changed or redefined during script execution. Unlike variables, constants are **immutable**.

php
Copy code

```
define("SITE_NAME", "My Website");
```

PHP Data Types

Variables in PHP are capable of storing various types of data, each categorized into eight distinct data types. These include basic, compound, and special data types, each serving a specific purpose. You have basic data types like Boolean, Integer, Double, and String. Additionally, PHP supports user-defined or compound data types, specifically Arrays and Objects. There are also special data types in PHP, namely NULL and resource.



This figure gives us a clear picture of PHP Data types.

Different Types of PHP Data Types

PHP has the following data types -

Under basic data types, we have -

- Integers
- Floats (Floating-Point Numbers)
- Strings
- Booleans
- PHP Compound Data Types
- Under compound data types, we have-
- Arrays
- Objects
- PHP Special Data Types
- Under special data types, we have-
- Resource
- NULL

PHP Integers

In PHP, integers are used to represent whole numbers, including both positive and negative values, without any fractional or decimal part. They can be expressed in three bases: decimal (base 10), octal (base 8), or hexadecimal (base 16), with the default base being decimal.

- Decimal Integers:
- $decimal_integer = 42;
- **Octal Integers**:
  $octal_integer = 052; // Leading 0 denotes octal
- **Hexadecimal Integers:**
  $hexadecimal_integer = 0x2A; // Leading 0x denotes hexadecimal

Example
This example demonstrates the PHP integer type.

```
<?php
$decimal_integer = 42;
$octal_integer = 052;
$hexadecimal_integer = 0x2a;
echo "The value of decimal_integer is: $decimal_integer<br/>";
echo "The value of octal_integer is: $octal_integer<br/>";
echo "The value of hexadecimal_integer is: $hexadecimal_integer<br/>";
// Printing values and types
var_dump($decimal_integer);
echo "<br/>";
var_dump($octal_integer);
echo "<br/>";
var_dump($hexadecimal_integer);
?>
```

Output:

```
The value of decimal_integer is: 42
The value of octal_integer is: 42
The value of Hexa decimal_integer is: 42
int(42)
int(42)
int(42)
```

PHP Floats

In PHP, floats are versatile data types that can represent numbers with fractional or decimal parts, including both positive and negative values. They can also express numbers in exponential form.

Decimal and Exponential Floats:

```
$decimal_float = 3.14;
$scientific_notation = 2.3e4; // 2.3 * 10^4
```

Example

This example demonstrates the PHP float data type.

```
<?php
$decimal_float = 3.14;
$scientific_notation = 2.3e4;
echo "The first value is: $decimal_float<br/>";
echo "The second value is: $scientific_notation.<br/>";
// Printing value and type
var_dump($decimal_float);
echo "<br/>";
var_dump($scientific_notation);
?>
```

Output:

```
The first value is: 3.14
The second value is: 23000.
float(3.14)
float(23000)
```

PHP Strings

Strings store sequences of characters, including letters and numbers. Use double quotes for flexibility and single quotes for literal values.

- **Double-Quoted String:** $double_quoted_string = "Hello, World!";
- **Single-Quoted String:** $single_quoted_string = 'PHP allows single-quoted strings.';

Note: Understanding the difference between single and double quotes is important, especially when dealing with variable values within strings.

Example

This example demonstrates the PHP string data type.

```
<?php
$str1 = 'Hello, world!';
$str2 = "Welcome at IncludeHelp";
echo "The first string is: $str1.<br/>";
echo "The second string is: $str2.<br/>";
// Printing value, size, and type
var_dump($str1);
echo "<br/>";
var_dump($str2);
?>
```

Output:

```
The first string is: Hello, world!.
The second string is: Welcome at IncludeHelp.
string(13) "Hello, world!"
string(22) "Welcome at IncludeHelp"
```

PHP Booleans

Booleans in PHP are fundamental for conditional testing, holding either TRUE (equivalent to 1) or FALSE (equivalent to 0).

Successful events typically return TRUE, while unsuccessful events return FALSE. Additionally:

- **NULL Values:**
  $null_value = NULL; // Treated as FALSE in boolean
- Zero Values:
  $zero_value = 0; // Also considered FALSE in boolean
- Empty Strings:
  $empty_string = ''; // Treated as FALSE in Boolean

Example

This example demonstrates the PHP Boolean data type.

```
<?php
$x = true;
if ($x) {
   echo "It's true.";
} else {
   echo "It's false.";
}
echo "<br/>";
$x = null;
if ($x) {
   echo "It's true.";
} else {
   echo "It's false.";
}
echo "<br/>";
$x = "Hello";
if ($x) {
   echo "It's true.";
} else {
   echo "It's false.";
}
?>
```

Output:

```
It's true.
It's false.
It's true.
```

PHP Arrays

Arrays, a compound data type in PHP, efficiently store multiple values of the same data type. They are like containers for holding multiple pieces of information. Here's an example with integers:

```
$integer_array = [1, 2, 3, 4, 5];
```

Example

This example demonstrates the PHP array data type.

```
<?php
$numbers = [1, 2, 3, 4, 5];
echo "First element: $numbers[0]<br/>";
echo "Second element: $numbers[1]<br/>";
echo "Third element: $numbers[2]<br/>";
echo "Fourth element: $numbers[3]<br/>";
echo "Fifth element: $numbers[4]<br/><br/>";
// Printing array's value and type
var_dump($numbers);
?>
```

Output:

```
First element: 1
Second element: 2
Third element: 3
Fourth element: 4
Fifth element: 5
array(5) { [0]=> int(1) [1]=> int(2) [2]=> int(3) [3]=> int(4) [4]=> int(5) }
```

PHP Objects

Objects in PHP are instances of user-defined classes. They encapsulate both data and functions specific to the class. Objects inherit properties and behaviors from the class, each instance having unique property values.

Example

```
<?php
class Car
{
   public $brand;
   public $model;
}
// Creating an instance of the Car class
$my_car = new Car();
$my_car->brand = "Toyota";
$my_car->model = "Camry";
echo $my_car->brand . "<br>";
echo $my_car->model;
?>
```

Output:
Toyota
Camry

In this example, we define a Car class with public properties $brand and $model. We then create an instance of the Car class named $my_car and set its properties to specific values. After creating the $my_car object, we demonstrate how to access its properties ($brand and $model) using the arrow (->) notation.

PHP NULL

NULL in PHP is a special variable type that can only hold one value: NULL. It is case-sensitive, typically written in capital letters. When a variable is created without a value, or explicitly set to NULL, it automatically takes on this special value. Let's understand NULL by this example.

Example

```
<?php
$variable = null;
echo "The variable is $variable <br>";

if (is_null($variable)) {
   echo "The variable is NULL";
}
?>
```

Output:

The variable is
The variable is NULL

In this example, we can see that the value of $variable is "NULL". It can be seen in the output that the first echo statement didn't display the $variable value.

PHP Resources

Resources in PHP are used to store references, often to external functions or resources, such as database connections. They are not an exact data type but serve as handles for specific operations like file handling or database connections.

Example

```
<?php
$integer_variable = 42;
$float_variable = 3.14;
$string_variable = "Hello, PHP!";
$boolean_variable = true;
$array_variable = [1, 2, 3];
$object_variable = new stdClass();
$null_variable = null;
echo "Integer: $integer_variable <br>";
echo "Float: $float_variable<br>";
echo "String: $string_variable<br>";
echo "Boolean: " . ($boolean_variable ? "true" : "false") . "<br>";
echo "Array: " . print_r($array_variable, true) . "<br>";
echo "Object: " . print_r($object_variable, true) . "<br>";
echo "NULL: " . var_export($null_variable, true) . "<br>";
?>
```

Output:

Integer: 42
Float: 3.14
String: Hello, PHP!
Boolean: true
Array: Array ([0] => 1 [1] => 2 [2] => 3)
Object: stdClass Object ( )

NULL: NULL

This example demonstrates the use of various PHP data types, including integers, floats, strings, booleans, arrays, objects and NULL.

Important Points on Variables and Constants in PHP

Variables in PHP

Dynamic Typing:

- PHP does not require explicit type declaration for variables. The type is determined by the value assigned.

Naming Rules:

- Variables must start with a dollar sign ($), followed by a letter or an underscore. They can contain letters, numbers, and underscores.

Global Scope:

- Variables declared outside of functions have global scope. Inside functions, global variables

must be accessed using the global keyword or $GLOBALS.

Super global:

- PHP has built-in global arrays like $_GET, $_POST, $_SESSION, and others that are accessible across the script.

Variable Variables:

- PHP allows the use of variable variables, where the name of a variable is stored in another variable.

References:

- Variables can be passed by reference using the & symbol, allowing functions to modify the original value.

Array Variables:

- PHP supports both indexed and associative arrays as variables to store multiple values.

Unset Variables:

- The unset() function is used to destroy a variable, making it unavailable in the current scope.

Default Values:

- You can use the ternary operator to set default values for variables when they are not defined.
- Constants in PHP

Definition:

- Constants are defined using the define() function or the const keyword in classes. They cannot be changed after being set.

Global Scope:

- Constants are automatically global and can be accessed anywhere in the PHP script.

Case Sensitivity:

- By default, constants are case-insensitive, but this can be controlled by passing a third argument in the define() function.

Magic Constants:

- PHP has built-in magic constants like __LINE__, __FILE__, __DIR__, etc., which provide contextual information about the script.

Constant Arrays:

- PHP allows defining constants as arrays (since PHP 5.6), enabling grouping of related constants.

Built-in Constants:

- PHP provides many built-in constants (e.g., PHP_VERSION, E_ALL, PHP_OS) that give information about the PHP environment.

Class Constants:

- Constants can be defined within classes using the const keyword, and can be accessed via ClassName::CONSTANT_NAME.

Difference from Variables:

- Constants cannot be modified once defined, unlike variables that can change values. They are often used for values that should remain unchanged throughout the script.

Practice Questions

1. What symbol is used to denote a variable in PHP?
A) % B) @ C) $ D) &

2.Which of the following is a valid way to declare a variable in PHP?
A) var name = "John"; B) $name = "John";
C) let name = "John"; D) define name = "John";

3. What will be the output of the following code: echo "Hello, $name"; if $name is set to "Alice"?
A) Hello, name B) Hello, Alice
C) Hello, $name D) Hello, "Alice"

4. Which of the following types can a PHP variable hold?
A) Only integers
B) Only strings
C) Multiple data types, including arrays and objects
D) Only booleans

5. What is the scope of a variable declared outside a function in PHP?
A) Local B) Global
C) Static D) Private

6. What is the correct way to define a constant in PHP?
A) const NAME = "Value";
B) define("NAME", "Value");
C) constant NAME = "Value";
D) variable NAME = "Value";

7. Which of the following statements about constants in PHP is true?
A) Constants can change their value after being defined.
B) Constants do not require a dollar sign ($).
C) Constants can only hold integer values.
D) Constants are case-insensitive by default.

8. What will happen if you try to redefine a constant in PHP?
A) It will change the value of the constant.
B) It will throw a fatal error.
C) It will ignore the new value.
D) It will redefine the constant with a warning.

9. What is the output of the following code?
define("SITE_NAME", "OpenAI");
echo SITE_NAME;
A) SITE_NAME B) OpenAI
C) "OpenAI" D) Error

10. Which function is used to check if a constant is defined in PHP?
A) is_constant() B) check_constant()
C) defined() D) constant_exists()

11. Which of the following is NOT a scalar data type in PHP?
A) Integer B) Float
C) Array D) String

12. What is the output of the following code?
$var = 10;
echo gettype($var);
A) Integer B) Float C) String D) Array

13. Which PHP data type is used to store multiple values in a single variable?
A) String B) Float
C) Array D) Boolean

14. How does PHP treat a variable that has not been initialized?
A) It automatically assigns it a value of 0.
B) It assigns it a NULL value.
C) It throws an error.
D) It assigns it a random value.

15. What will be the output of the following code?
$var = "5";
$result = $var + 2;
echo $result;
A) 5 B) 7 C) 52 D) Error

16. Which of the following is an example of an associative array in PHP?
A) $colors = array("red", "green", "blue");
B) $ages = array("Alice" => 30, "Bob" => 25);
C) $numbers = array(1, 2, 3);
D) $data = array(10, "text", true);

17. What is the purpose of the isset() function in PHP?
A) To check if a variable is defined and is not NULL.
B) To check if a variable is an array.
C) To retrieve the type of a variable.
D) To count the number of elements in an array.

18. Which of the following data types can represent both true and false values in PHP?
A) Integer B) String
C) Boolean D) Float

19. What will be the result of the following code?
$var = "Hello";
$var .= " World!";
echo $var;
A) Hello B) Hello World!
C) HelloWorld! D) Error

20. Which PHP function can be used to convert a string to an integer?
A) to_integer() B) intval()
C) str_to_int() D) cast_int()

Ans : 1-c,2-b,3-b,4-c,5-b,6-b,7-b,8-b,9-b,10-c,11-c,12-a,13-c,14-b,15-b,16-b,17-a,18-c,19-b,20-b

Professor Academy

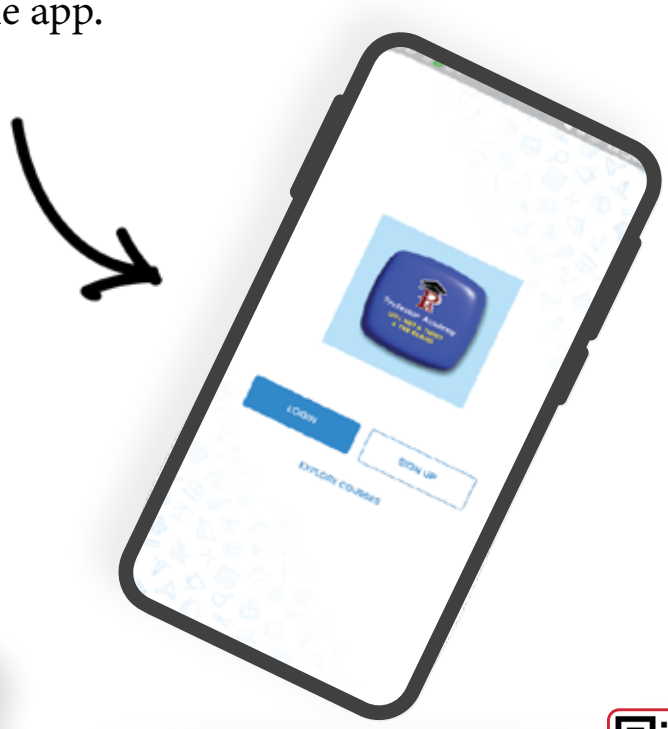
TEST SERIES

Extensive Test Series to Boost Your Preparation!

Get access to 100+ Test Series, including:

- **Daily Tests** to reinforce learning every day
- **Unit-Wise Tests** for focused revision
- **Full-Length Tests** to simulate the real exam experience

Comprehensive Study Materials as soft copy PDFs, available in our mobile app.



Download & Explore!

www.professoracademy.com

Professor Academy

STUDY MATERIALS

Enroll today and receive 12 Printed Books right at your doorstep

- **10 Subject-Based Books** covering all major topics
- **1 Tamil Eligibility Book**
- **1 Previous Year Question Bank**

(Subject Book + PYQ Bank available only in English Medium)

Comprehensive Study Materials as soft copy PDFs, available in our mobile app.

PG TRB COMPUTER SCIENCE

Based on New Syllabus

Rs. 300 for courier charge payments is excluded

www.professoracademy.com

3. Operators and Expressions

PHP Operators

PHP operators are characters or sets of characters used to manipulate or perform operations on expressions and values. Operators allow you to perform arithmetic operations, assign values to variables, string concatenation, compare deals, and perform boolean operations.

Operators are fundamental building blocks of programming languages. They allow us to manipulate and combine data to achieve desired results. In PHP, there are many operators available for various purposes. This tutorial will introduce you to the most common PHP operators and provide examples of their usage.

The following are the operators in PHP:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Increment/Decrement Operators
5. Logical Operators
6. String Operators
7. Array Operators
8. Conditional Assignment Operators

PHP Arithmetic Operators

Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication, and division.

The following are the PHP arithmetic operators:

Operator	Description	Example
+	Addition	$sum = 100 + 50; // $sum will be 150
-	Subtraction	$difference = 200 - 100; // $difference will be 100
*	Multiplication	$product = 5 * 3; // $product will be 15
/	Division	$quotient = 100 / 2; // $quotient will be 50
%	Modulus	$remainder = 100 % 3; // $remainder will be 1

Example of PHP Arithmetic Operators

```
<?php
$sum = 100 + 50;
$difference = 200 - 100;
$product = 5 * 3;
$quotient = 100 / 2;
$remainder = 100 % 3;
// printing
echo $sum . "<br>";
echo $difference . "<br>";
echo $product . "<br>";
echo $quotient . "<br>";
echo $remainder . "<br>";
?>
```

The output of the above example is:

150
100
15
50
1

PHP Assignment Operators

Assignment operators are used to assign values to variables.

The following are the PHP assignment operators:

Operator	Description	Example
=	Simple assignment	$a = 65;
+=	Adds and assigns	$a += 5; means (a=a+5) // $a will be 70
-=	Subtracts and assigns	$a -= 5; means (a=a-5) // $a will be 65
*=	Multiplies and assigns	$a *= 2; means (a*2) // $a will be 130
/=	Divides and assigns	$a /= 2; // $a will be 10

Example of PHP Assignment Operators

```
<?php
$a = 65;
echo $a . "<br>";
$a += 5;
echo $a . "<br>";
$a -= 5;
echo $a . "<br>";
$a *= 2;
echo $a . "<br>";
$a /= 2;
echo $a . "<br>";
?>
```

The output of the above example is:

```
65
70
65
130
65
```

PHP Comparison Operators

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

The following are the PHP comparison operators:

Operator	Description	Example
==	Equal to	$a == 10; // true
!=	Not equal to	$a != 10; // false
<	Less than	$a < 10; // false
>	Greater than	$a > 10; // false
<=	Less than or equal to	$a <= 10; // true
>=	Greater than or equal to	$a >= 10; // true
<=>	Spaceship operator (PHP 7+)	$a <=> 10; // -1

Example of PHP Comparison Operators

```
<?php
$a = 10;
$result = $a == 10;
var_dump($result);
echo "<br>";
$result = $a != 10;
var_dump($result);
echo "<br>";
$result = $a < 10;
var_dump($result);
echo "<br>";
$result = $a > 10;
var_dump($result);
echo "<br>";
$result = $a <= 10;
var_dump($result);
echo "<br>";
$result = $a >= 10;
var_dump($result);
echo "<br>";
$result = $a <=> 10;
var_dump($result);
echo "<br>";
?>
```

The output of the above example is:

```
bool(true)
bool(false)
bool(false)
bool(false)
bool(true)
bool(true)
int(0)
```

PHP Increment/Decrement Operators

Increment and decrement operators are used to increase or decrease the value of a variable by 1.

The following are the PHP increment/decrement operators:

Operator	Description	Example
++	Pre-increment	++$a; // $a becomes 11
--	Pre-decrement	--$a; // $a becomes 9
$a++	Post-increment	$a++; // $a becomes 10 (after the statement execution)
$a--	Post-decrement	$a--; // $a becomes 9 (after the statement execution)

Example of PHP Increment/Decrement Operators

```
<?php
$a = 10;
++$a;
echo $a . "<br>";
--$a;
echo $a . "<br>";
$a++;
echo $a . "<br>";
$a--;
echo $a . "<br>";
?>
```

The output of the above example is:

```
11
10
11
10
```

PHP Logical Operators

Logical operators are used to combine two or more Boolean values and return a single Boolean result.

The following are the PHP logical operators:

Operator	Description	Example
&&	Logical And	$a > 10 && $b < 5; // true only if both $a > 10 and $b < 5 are true
||	Logical Or	$a and $b; // Return true if both are true
!	Logical Not	!($a > 10); // true if $a is not greater than 10
and	Logical And	$a and $b; // Return true if both are true
or	Logical Or	$a or $b; // Return true if either is true
xor	Logical Xor	$a xor $b; // Return true if either is true, but not both

Example of PHP Logical Operators

```
<?php
$a = 10;
$b = 5;
$result = $a == 10 && $b == 5;
var_dump($result);
echo "<br>";
$result = $a == 10 || $b == 5;
var_dump($result);
echo "<br>";
$result = !($a == 10 && $b == 5);
var_dump($result);
echo "<br>";
$result = ($a == 10 and $b == 5);
var_dump($result);
echo "<br>";
$result = ($a == 10 or $b == 5);
var_dump($result);
echo "<br>";
$result = ($a == 10 xor $b == 5);
var_dump($result);
echo "<br>";
?>
```

The output of the above example is:

```
bool(true)
bool(true)
bool(false)
bool(true)
bool(true)
bool(false)
```

PHP String Operators

String operators are used to manipulate and combine strings.

The following are the PHP string operators:

Operator	Description	Example
.	Concatenation	$str1 = "Include" . " " . $str2;
.=	String concatenation and assignment	$str1 .= "Help";
==	Equal to	$str1 == "Include Help"; // true
!=	Not equal to	$str1 != "Include Help"; // false

Example of PHP String Operators

```
<?php
$str1 = "Include";
$str2 = "Help";
$result = $str1 . $str2;
echo $result . "<br>";
$result .= $str1;
echo $result . "<br>";
$result = $str1 == $str2;
var_dump($result);
echo "<br>";
$result = $str1 != $str2;
var_dump($result);
echo "<br>";
?>
```

The output of the above example is:

```
IncludeHelp
IncludeHelpInclude
bool(false)
bool(true)
```

PHP Array Operators

Array operators in PHP are used to compare two arrays, these are basically a kind of relation operators for the array companions.

The following are the PHP array operators:

Operator	Description	Example
+	Array Union	$a + $b; - Returns union or arrays.
==	Array Equality	$a == $b; - Returns if both have the same key and value pairs.
===	Array Identity	$a == $b; - Returns if both have the same key and value pairs (their order and type must be same).
!=	Array Inequality	$a != $b; - Returns true if both are not equal.
<>	Array Inequality	$a <> $b; - Returns true if both are not equal.
!==	Array non-identity	$a <> $b; - Returns true if $a is not identical to $b.

Example of PHP Array Operators

```
<?php
$x = ["fruit" => "Apple", "color" => "Red"];
$y = ["company" => "Honda", "model" => "City"];
print_r($x + $y);
echo "<br>";
var_dump($x == $y);
echo "<br>";
var_dump($x === $y);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x <> $y);
echo "<br>";
var_dump($x !== $y);
echo "<br>";
?>
```

The output of the above example is:

```
Array ( [fruit] => Apple [color] => Red [company] => Honda [model] => City )
bool(false)
bool(false)
bool(true)
bool(true)
bool(true)
```

PHP Conditional Assignment Operators

Conditional assignment operators are used to define value based on the given conditions.

The following are the PHP conditional assignment operators:

Operator	Description	Example
?:	Ternary Operator	$result = expr1 ? expr2 : expr2;
??	Non-Coalescing Operator	$result = expr1 ?? expr2;

Example of PHP Conditional Assignment Operators

```
<?php
$a = 10;
$b = 5;
$result = $a > $b ? $a : $b;
echo "Largest value is " . $result . "<br>";
// Assigns "Hello, world!",
// if $welcome either not defined or null
$welcome = $welcome ?? "Hello, world!";
echo "Welcome message is " . $welcome . "<br>";
?>
```

The output of the above example is:

```
Largest value is 10
Welcome message is Hello, world!
```

PHP Operators Practice (More Examples)

Exercise 1

Write a PHP script that calculates the area of a rectangle given its length and width using arithmetic operators.

```
<?php
// Calculate the area of a rectangle
$length = 5;
$width = 8;
$area = $length * $width;
echo "The area of the rectangle is: $area";
?>
```

The output of the above example is:

The area of the rectangle is: 40

Exercise 2

Create a PHP script that checks if a given number is both divisible by 2 and 3. Use logical operators to perform the check.

```
<?php
// Check if a number is divisible by 2 and 3
$number = 12;
if ($number % 2 == 0 && $number % 3 == 0) {
   echo "$number is divisible by both 2 and 3.";
} else {
   echo "$number is not divisible by both 2 and 3.";
}
?>
```

The output of the above example is:

12 is divisible by both 2 and 3.

Exercise 3

Write a PHP function that compares two strings and returns "Equal" if they are the same, "Greater" if the first string is lexicographically greater, and "Smaller" if the second string is lexicographically greater.

```
<?php
// Compare two strings
function compareStrings($str1, $str2)
{
   $result = strcmp($str1, $str2);
   if ($result == 0) {
  return "Equal";
   } elseif ($result > 0) {
  return "Greater";
   } else {
  return "Smaller";
   }
}
echo compareStrings("apple", "orange");
?>
```

The output of the above example is:

Smaller

Exercise 4

Implement a PHP script that determines whether a given number is positive, negative, or zero using the ternary operator.

```
<?php
// Determine if a number is positive, negative, or zero
$number = -7;
$result = $number > 0 ? "Positive" : ($number < 0 ? "Negative" : "Zero");
echo "The number is $result.";
?>
```

The output of the above example is:

The number is Negative.

Exercise 5

Develop a PHP program that uses the multiplication and assignment operator (*=) to calculate the square of a number.

```
<?php
// Calculate the square of a number using *=
$number = 4;
$number *= $number;
echo "The square of the number is: $number";
?>
```

The output of the above example is:

The square of the number is: 16

Exercise 6

Write a PHP function that takes two numbers as parameters and uses the spaceship operator to return:

- -1 if the first number is less than the second,
- 0 if they are equal,
- 1 if the first number is greater than the second.

```
<?php
```

```
// Use spaceship operator to compare two numbers
function compareNumbers($num1, $num2)
{
   return $num1 <=> $num2;
}
$result = compareNumbers(8, 5);
echo "Result: $result";
?>
```

The output of the above example is:

Result: 1

Exercise 7

Create an array of fruits and another array of vegetables. Use array operators to merge these arrays and display the combined list.

```
<?php
// Merge arrays of fruits and vegetables
$fruits = ["apple", "banana", "orange"];
$vegetables = ["carrot", "broccoli", "spinach"];
$combined = $fruits + $vegetables;
print_r($combined);
?>
```

The output of the above example is:

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

Exercise 8

Implement a PHP script that uses a loop to print the numbers from 1 to 10 using the post-increment operator.

```
<?php
// Print numbers from 1 to 10 using post-increment operator
$i = 1;
while ($i <= 10) {
   echo "$i ";
   $i++;
}
?>
```

The output of the above example is:

1 2 3 4 5 6 7 8 9 10

Exercise 9

Write a PHP function that takes two strings as parameters and concatenates them using the concatenation and assignment operator (.=). Return the resulting string.

```
<?php
// Concatenate two strings using .=
function concatenateStrings($str1, $str2)
{
   $str1 .= $str2;
   return $str1;
}
echo concatenateStrings("Hello", " World!");
?>
```

The output of the above example is:

Hello World!

PHP Expressions

An expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a value.

In PHP, an expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a single value. These elements come together to form a logical operation, and the result of this operation is the output of the expression.

Components of an Expression:

- Values: These are the raw data elements in programming. Examples include numbers (3, 10.2), strings (‘Hello World'), and booleans (true, false).
- Variables: Variables are containers for holding values. They have names and can store different values at different points in program’s execution.
- Operators: Operators are symbols or keywords that perform operations on values and variables. Examples include arithmetic operators (+, -, * /), comparison operators (==, !=, <, >) and logical operators (&&, ||).
- Function Calls: Functions are reusable blocks of code that perform specific tasks. When you call a function, it can return a value. Function calls typically involve passing arguments enclosed in parentheses. Visit @pr0fess0racademy.com

Exam Points

PHP expressions are evaluated based on the **precedence and associativity** of the operators involved. PHP follows a standard set of rules to determine which operations to perform first. For example, multiplication has a higher precedence than addition.

$result = 5 + 3 * 2; // Outputs 11, not 16, because multiplication is done first

Examples of Expressions

Basic Expressions

The simplest forms of expressions are constants and variables.

For example:

```
$a = 5; // 5 is an expression with the value 5
$b = $a; // $a is an expression with the value 5
```

Functions are also expressions. For instance:

```
function foo() {
return 5;
}
$c = foo(); // foo() is an expression with the value 5
```

PHP supports four scalar value types: int, float, string, and bool. It also supports two composite types: arrays and objects

Arithmetic Expression:

$result = 5 + 3;

Here, the expression 5+3 adds the values 5and 3, resulting in the value 8.

String Concatenation Expression:

$name = 'John';

$greetings = 'Hello', . $name;

The expression "Hello, " . $name concatenates the string "Hello, " with the value store in the variable $name .

Function Call Expression:

$length = strlen('Hello');

The expression calls the strlen function to get the length of the string 'Hello' , producing the value 5 .

Comparison Expression:

Comparison expressions evaluate to either true or false. PHP supports various comparison operators:

```
$a > $b; // greater than
$a >= $b; // greater than or equal to
$a == $b; // equal
$a != $b; // not equal
$a < $b; // less than
$a <= $b; // less than or equal to
$a === $b; // identical (equal and same type)
$a !== $b; // not identical (not equal or not same type)
```

Ternary Operator Expression:

$number = 7;

$isEven = ($number % 2 == 0) ? true : false;

The expression uses the ternary operator to determine if. $number is even, producing true or false accordingly.

Increment and Decrement Operators

PHP supports pre-increment (++$variable) and post-increment ($variable++) operators. The difference lies in the value of the increment expression: These operators are called increment and decrement operators respectively. They are unary operators, needing just one operand and can be used in prefix or postfix manner, although with different effect on value of expression.

Both prefix and postfix ++ operators increment value of operand by 1 (whereas -- operator decrements by 1). However, when used in assignment expression, prefix makes increment/decrement first and then followed by assignment. In case of postfix, assignment is done before increment/decrement

Uses postfix ++ operator

Example

```
<?php
$x=10;
$y=$x++; //equivalent to $y=$x followed by $x=$x+1
echo "x = $x y = $y";
?>
```

Output

This produces following result

x = 11 y = 10

Whereas following example uses prefix increment operator in assignment

Example

```
<?php
$x=10;
```

$y=++$x;; //equivalent to $x=$x+1 followed by $y=$x
echo "x = $x y = $y";
?>

Output

x = 11 y = 11

Pre-increment increments the variable before reading its value, while post-increment increments 1the variable after reading its value[1].

Assignment Expressions

Assignments in PHP are expressions that evaluate to the assigned value. **For example:**

$a = 5; // $a = 5 is an expression with the value 5
$b = ($a = 5); // $b = $a = 5 is like writing $a = 5; $b = 5;

Assignments are parsed from right to left, so you can write:

$b = $a = 5;

Practice Questions

1. What will the following PHP expression output? echo 5 + 10 * 2;
A) 15 B) 25 C) 20 D) 10

2.Which operator is used for string concatenation in PHP?
A) . B) + C) & D) *

3. What will the value of $x be after executing $x = 5; $x += 10;?
A) 15 B) 10 C) 5 D) 0

4. What is the result of the following expression: true && false?
A) true B) false C) null D) 1

5. Which of the following will compare two values for equality without type conversion?
A) == B) === C) != D) !==

6. What does the ?? operator do in PHP?
A) Returns the first operand if it's true
B) Returns the first non-null operand
C) Concatenates two strings
D) Compares two values

7. What will the output of the following code be?

```
php
$a = 1;
$b = '1';
if ($a == $B)  {
   echo "Equal";
} else {
   echo "Not Equal";
}
```

A) Equal B) Not Equal
C) Error D) Undefined

8. Which operator would you use to increment a variable by one?
A) ++ B) +1 C) +=1 D) --

9. In PHP, what is the result of 10 % 3?
A) 3 B) 1 C) 0 D) 10

10. What will isset($var) return if $var is not defined?
A) true B) false
C) null D) undefined

11. What will the expression !(true || false) evaluate to?
A) true B) false C) null D) 1

12. Which operator is used to check if a variable is not set or is null?
A) isset() B) empty() C) ! D) ??

13. What will the result of the expression 5 ** 2 be in PHP?
A) 10 B) 25 C) 2.5 D) 5

14. If $a = 5; $b = 3; what will $a <=> $b evaluate to?
A) -1 B) 0 C) 1 D) null

15. What will the output of echo 10 == "10"; be?
A) true B) false C) 1 D) Error

16. Which of the following operators is not a bitwise operator?
A) & B) | C) ^ D) &&

17. What will the following code output?

```
$x = 10;
$y = 20;
$result = $x < $y ? 'Less' : 'Greater';
echo $result;
```

A) Less B) Greater
C) Error D) 10

18. What is the outcome of array(1, 2, 3) + array(4, 5, 6)?
A) array(1, 2, 3, 4, 5, 6)
B) array(1, 2, 3)
C) Error
D) array(0 => 1, 1 => 2, 2 => 3)

19. Which of the following is the correct way to concatenate two strings in PHP?
A) $str1 & $str2 B) $str1 + $str2
C) $str1 . $str2 D) $str1 * $str2

20. What will the result of 5 == '5' be in PHP?
A) true B) false C) 1 D) 0

Ans : 1-b,2-a,3-a,4-b,5-b,6-b,7-a,8-a,9-b,10-b,11-b,12-d,13-b,14-c,15-a,16-d,17-a,18-b,19-c,20-a

Professor Academy

PAYMENT DETAILS& ENROLLMENT PROCESS

BOOKS ONLY	COURSE ONLY	COURSE + BOOKS
₹3999	₹8999	₹11999
ACTUAL FEE ~~₹5999~~	ACTUAL FEE ~~₹14999~~	ACTUAL FEE ~~₹19999~~

An additional charge of Rs. 300 for courier charge payments is excluded in the original course fee.

Step 1: Register Online on Our Website or App.

SCAN HERE
TO REGISTER

Step 2: Choose Your Course and Complete Payment

Step 3: Receive Login Details and Begin Your Journey!

Step 4: Our dedicated support team is here to assist you at every step.

NOTE* Books will be provided only after the full payment of fees has been made.

www.professoracademy.com

Professor Academy

Professor Academy's Pride
TRB State Rankers

ISWARYA P

DEVASAGAYAM D

RAJESHWARI N

Achiever's Testimonials

Line by line teaching helps me to take good notes. They teach me lessons thoroughly, and then only I attend their tests. Their test makes me more focused on preparation.

VINITHA. M | STATE 6TH RANK

Professor Academy's test series are good. I write each and every test like TRB exam. I am very happy to attend Professor Academy's test. Happy to see my test result. In a similar way, I secured state rank on PG TRB exam (just assume exam like Professor Academy's test).

AMUTHA VIJAYALAKSHMI. A | STATE 7TH RANK

Professor Academy's motivation and their notes are very helpful for me. Their crystal-clear explaining faculties are the secret of my success. Their friendly guidance boosts my exam preparation.

HEMA. P | STATE 8TH RANK

The reason for my success is Professor Academy. If not for Professor Academy, I wouldn't have achieved this. The recorded classes are very useful for me. It made studying easier and gave me confidence. Therefore, I recommend Professor Academy to everyone who wants to succeed in their studies.

ARUNA RAJESWARI. R | STATE 10TH RANK

www.professoracademy.com

Click here to Watch more Achiever's Talks

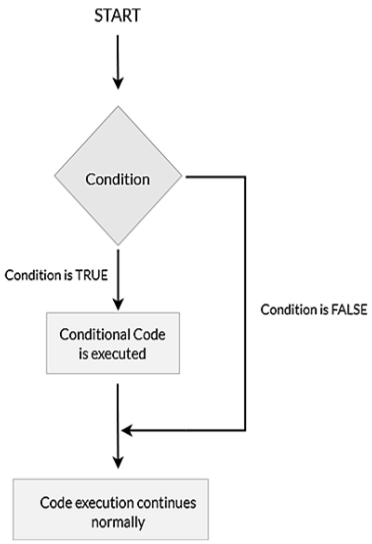
4. Control Structures in PHP

Introduction

In simple terms, a control structure allows you to control the flow of code execution in your application. Generally, a program is executed sequentially, line by line, and a control structure allows you to alter that flow, usually depending on certain conditions.

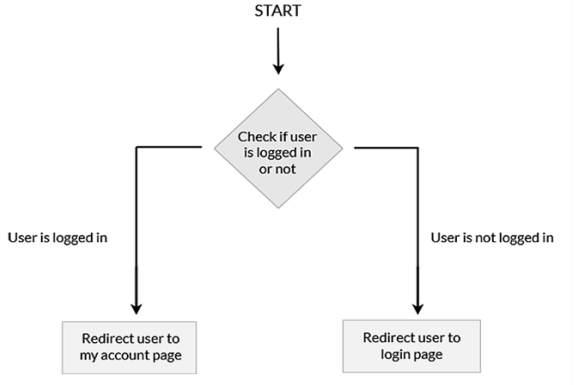
Control structures are core features of the PHP language that allow your script to respond differently to different inputs or situations. This could allow your script to give different responses based on user input, file contents, or some other data.

The following flowchart explains how a control structure works in PHP.



First a condition is checked. If the condition is true, the conditional code will be executed. The important thing to note here is that code execution continues normally after conditional code execution.

Let's consider the following example



In the above example, the program checks whether or not the user is logged in. Based on the user's login status, they will be redirected to either the Login page or the My Account page. In this case, a control structure ends code execution by redirecting users to a different page. This is a crucial ability of the PHP language.

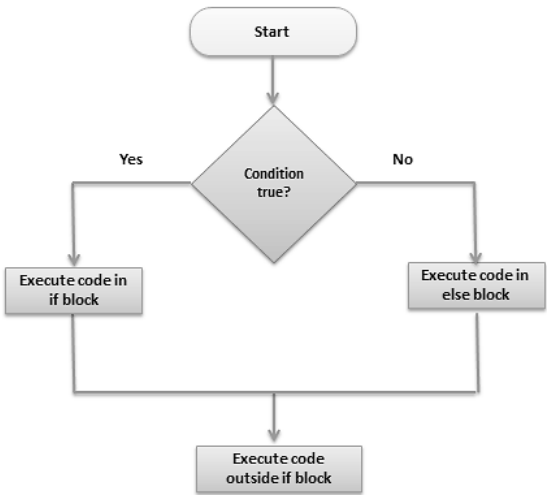
Different types of control structures:

PHP supports a number of different control structures:

- if
- else
- elseif
- switch
- while
- do-while
- for
- foreach and more

PHP If Statement

The if construct allows you to execute a piece of code if the expression provided along with it evaluates to true.



Syntax

The syntax for if… then… else is;

```
<?php
if (condition is true) {
block one
else
block two
}
```

```
?>
```

Let's have a look at the following example to understand how it actually works.

The code below uses “if… then… else” to determine the larger value between two numbers.

```
<?php

$first_number = 7;
$second_number = 21;
if ($first_number > $second_number){
echo "$first_number is greater than $second_number";
}else{
echo "$second_number is greater than $first_number";
}
?>
```

Output:

21 is greater than 7

PHP Else Statement

In the previous section, we discussed the if construct, which allows you to execute a piece of code if the expression evaluates to true. On the other hand, if the expression evaluates to false, it won't do anything. More often than not, you also want to execute a different code snippet if the expression evaluates to false. That's where the else statement comes into the picture.

You always use the else statement in conjunction with an if statement. Basically, you can define it as shown in the following pseudo-code.

```
if (expression)
{
  // code is executed if the expression evaluates to TRUE
}
else
{
  // code is executed if the expression evaluates to FALSE
}
```

Let's revise the previous example to understand how it works.

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

Output:

Your age is greater than or equal to 30!

So when you have two choices, and one of them must be executed, you can use the if-else construct.

PHP Else If Statement

We can consider the elseif statement as an extension to the if-else construct. If you've got more than two choices to choose from, you can use the elseif statement.

Let's study the basic structure of the elseif statement, as shown in the following pseudo-code.

```
if (expression1)
{
  // code is executed if the expression1 evaluates to TRUE
}
elseif (expression2)
{
  // code is executed if the expression2 evaluates to TRUE
}
elseif (expression3)
{
  // code is executed if the expression3 evaluates to TRUE
}
else
{
  // code is executed if the expression1, expression2 and expression3 evaluates to FALSE, a default choice
}
```

Again, let's try to understand it using a real-world example.

```
<?php
```

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

Output

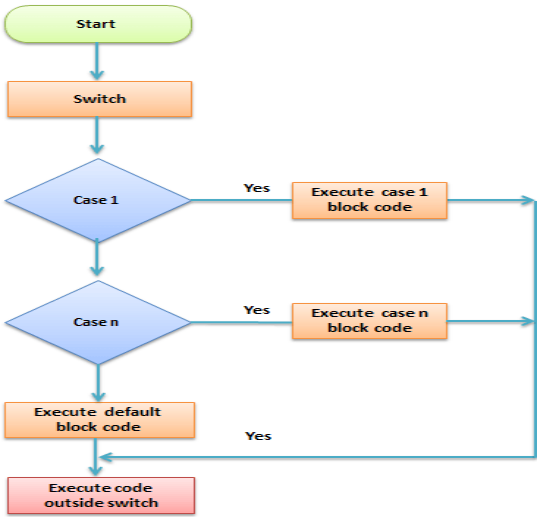
Your age is greater than 50!

As you can see in the above example, we have multiple conditions, so we've used a series of elseif statements. In the event that all if conditions evaluate to false, it executes the code provided in the last else statement.

PHP Switch Statement

The switch statement is somewhat similar to the elseif statement which we've just discussed in the previous section. The only difference is the expression which is being checked.

In the case of the elseif statement, you have a set of different conditions, and an appropriate action will be executed based on a condition. On the other hand, if you want to compare a variable with different values, you can use the switch statement.



Exam Points

The **break** statement is used inside **switch** (and other loops) to terminate the loop or switch case early. Without break, the program will continue checking the subsequent cases, which may not be desirable.

As usual, an example is the best way to understand the switch statement.

```
<?php
$favourite_site = 'Code';
switch ($favourite_site) {
  case 'Business':
    echo "My favourite site is business.tutsplus.com!";
    break;
  case 'Code':
    echo "My favourite site is code.tutsplus.com!";
    break;
  case 'Web Design':
    echo "My favourite site is webdesign.tutsplus.com!";
    break;
  case 'Music':
    echo "My favourite site is music.tutsplus.com!";
    break;
  case 'Photography':
    echo "My favourite site is photography.tutsplus.com!";
    break;
  default:
    echo "I like everything at tutsplus.com!";
}
?>
```

As you can see in the above example, we want to check the value of the $favourite_site variable, and based on the value of the $favourite_site variable, we want to print a message.

For each value you want to check with the $favourite_site variable, you have to define the case block. If the value is matched with a case, the code associated with that case block will be executed. After that, you need to use the break statement to end code execution.

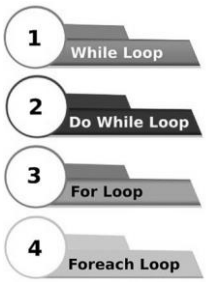
If you don't use the break statement, script execution will be continued up to the last block in the switch statement.

Finally, if you want to execute a piece of code if the variable's value doesn't match any case, you can define it under the default block. Of course, it's not mandatory—it's just a way to provide a default case.

So that's the story of conditional control structures. We'll discuss loops in PHP in the next section.

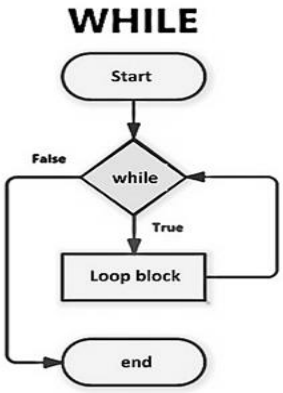
Loops in PHP

Loops in PHP are useful when you want to execute a piece of code repeatedly until a condition evaluates to false. So code is executed repeatedly as long as a condition evaluates to true, and as soon as the condition evaluates to false, the script continues executing the code after the loop.



While Loop in PHP

The while loop is used when you want to execute a piece of code repeatedly until the while condition evaluates to false.



You can define it as shown in the following pseudo-code.

```
while (expression)
{
 // code to execute as long as expression evaluates to TRUE
}
```

Let's have a look at a real-world example to understand how the while loop works in PHP.

```
<?php
$max = 0;
echo $i = 0;
echo ",";
echo $j = 1;
echo ",";
$result=0;
while ($max < 10 )
{
   $result = $i + $j;
   $i = $j;
   $j = $result;
   $max = $max + 1;
   echo $result;
   echo ",";
}
?>
```

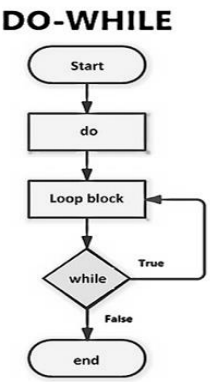
Output

0,1,1,2,3,5,8,13,21,34,55,89,

If you're familiar with the Fibonacci series, you might recognize what the above program does—it outputs the Fibonacci series for the first ten numbers. The while loop is generally used when you don't know the number of iterations that are going to take place in a loop.

Do-While Loop in PHP

The do-while loop is very similar to the while loop, with the only difference being that the while condition is checked at the end of the first iteration. Thus, we can guarantee that the loop code is executed at least once, irrespective of the result of the while expression.



syntax of the do-while loop.

```
do
{
  // code to execute
} while (expression);
```

Let's go through a real-world to understand possible cases where you can use the do-while loop.

```
<?php
$handle = fopen("file.txt", "r");
if ($handle)
{
   do
   {
  $line = fgets($handle);
  // process the line content
   } while($line !== false);
}
fclose($handle);
?>
```

In the above example, we're trying to read a file line by line. Firstly, we've opened a file for reading. In our case, we're not sure if the file contains any content at all. Thus, we need to execute the fgets function at least once to check if a file contains any content. So we can use the do-while loop here. do-while evaluates the condition after the first iteration of the loop.

For Loop in PHP

Generally, the for loop is used to execute a piece of code a specific number of times. In other words, if you already know the number of times you want to execute a block of code, it's the for loop which is the best choice.

Let's have a look at the syntax of the for loop

```
for (expr1; expr2; expr3)
{
  // code to execute
}
```

The expr1 expression is used to initialize variables, and it's always executed. The expr2 expression is also executed at the beginning of a loop, and if it evaluates to true, the loop code is executed. After execution of the loop code, the expr3 is executed. Generally, the expr3 is used to alter the value of a variable which is used in the expr2 expression.

Let's go through the following example to see how it works.

```
<?php
for ($i=1; $i<=10; ++$i)
{
  echo sprintf("The square of %d is %d.</br>", $i, $i*$i);
}
?>
```

The above program outputs the square of the first ten numbers. It initializes $i to 1, repeats as long as $i is less than or equal to 10, and adds 1 to $i at each iteration.

For Each in PHP

The foreach loop is used to iterate over array variables. If you have an array variable, and you want to go through each element of that array, the foreach loop is the best choice.

Let's have a look at a couple of examples.

```
<?php
$fruits = array('apple', 'banana', 'orange', 'grapes');
foreach ($fruits as $fruit)
{
  echo $fruit;
  echo "<br/>";
}
$employee = array('name' => 'John Smith', 'age' => 30, 'profession' => 'Software Engineer');
foreach ($employee as $key => $value)
{
  echo sprintf("%s: %s</br>", $key, $value);
  echo "<br/>";
}
?>
```

If you want to access array values, you can use the first version of the foreach loop, as shown in the above example. On the other hand, if you want to access both a key and a value, you can do it as shown in the $employee example above.

Breaking Out of the Loop

There are times when you might want to break out of a loop before it runs its course. This can be achieved easily using the break keyword. It will get you out of the current for, foreach, while, do-while, or switch structure.

You can also use break to get out of multiple nested loops by supplying a numeric argument. For example, using break 3 will break you out of 3 nested loops. However, you cannot pass a variable as the numeric argument if you are using a PHP version greater than or equal to 5.4

```
<?php
echo "Simple Break\n";
for($i = 1; $i <= 2; $i++) {
    echo "$i = $i ";
    for($j = 1; $j <= 5; $j++) {
   if($j == 2) {
  break;
   }
   echo "$j = $j ";
    }
    echo "\n";
}

echo "Multi-level Break\n";
for($i = 1; $i <= 2; $i++) {
    echo "$i = $i ";
    for($j = 1; $j <= 5; $j++) {
   if($j == 2) {
  break 2;
   }
   echo "$j = $j ";
    }
    echo "\n";
}
Output
Simple Break
1 = 1 1 = 1
2 = 2 2 = 1

Multi-level Break
1 = 1 1 = 1
```

Another keyword that can interrupt loops in PHP is continue. However, this only skips the rest of the current loop iteration instead of breaking out of the loop altogether. Just like break, you can also use a numerical value with continue to specify how many nested loops it should skip for the current iteration.

```
<?php
echo 'Simple Continue';
for($i = 1; $i <= 2; $i++) {
    echo "\n".'$i = '.$i.' ';
    for($j = 1; $j <= 5; $j++) {
   if($j == 2) {
  continue;
   }
   echo '$j = '.$j.' ';
    }
}
/*
Simple Continue
i = 1 j = 1 j = 3 j = 4 j = 5
i = 2 j = 1 j = 3 j = 4 j = 5
*/
echo 'Multi-level Continue';
for($i = 1; $i <= 2; $i++) {
    echo "\n".'$i = '.$i.' ';
    for($j = 1; $j <= 5; $j++) {
   if($j == 2) {
  continue 2;
   }
   echo '$j = '.$j.' ';
    }
}
/*
Multi-level Continue
i = 1 j = 1
i = 2 j = 1
*/
?>
```

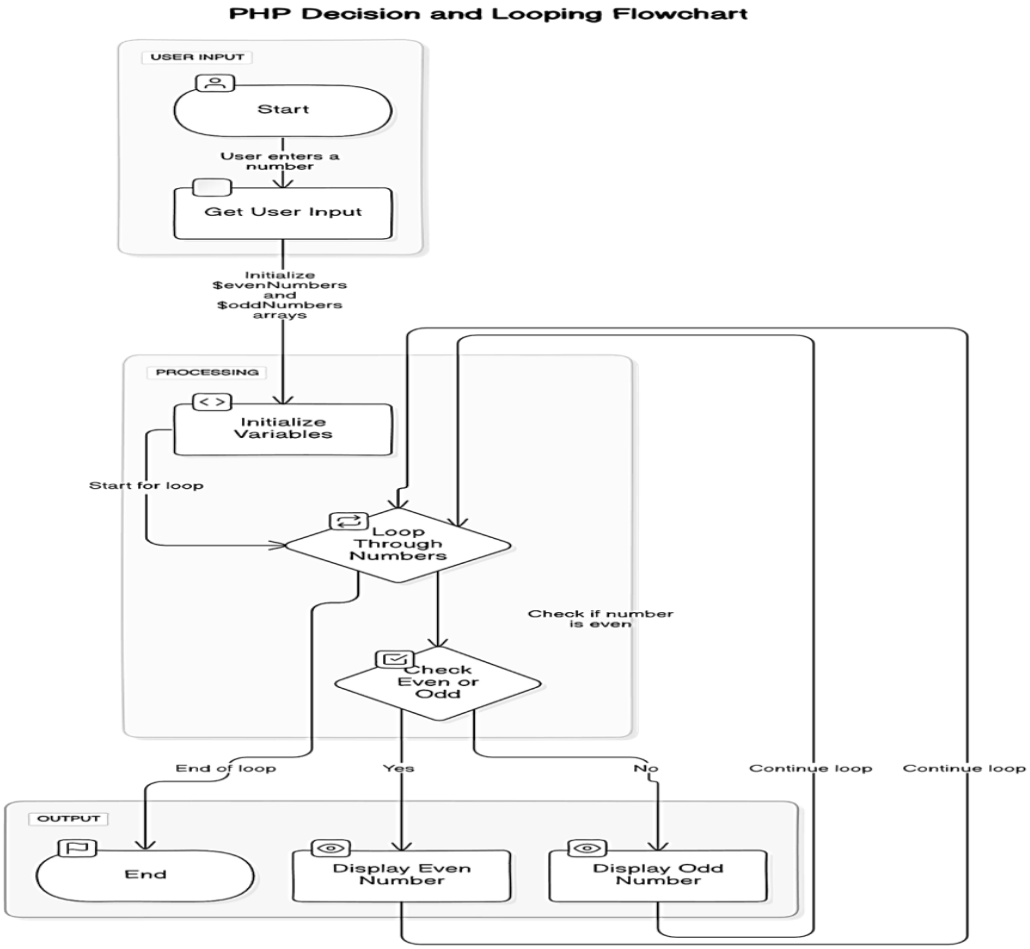
Exam Points

The **goto** statement in PHP allows you to jump to another part of the program. It's often discouraged, as it can lead to code that's difficult to understand and maintain, but it can be useful in specific cases.

```
<php>
goto mylabel;
echo "This will be skipped.";
mylabel:
echo "This will be executed.";
<?php>
```

PHP Mixing Decisions and looping with Html

Combining of decisions (if statement) and looping (foreach loop) in PHP within an HTML document. In this example, we'll use PHP to list of even and odd numbers based on user input:



Here's a simple example that demonstrates the use of decisions (if statement) and looping (for loop) within an HTML document:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PHP Decision and Looping Example</title>
</head>
<body>
  <h1>PHP Decision and Looping Example</h1>
  <form action="" method="post">
  <label for="userInput">Enter a number:</label>
  <input type="number" name="userInput" id="userInput" required>
  <input type="submit" value="Generate Numbers">
  </form>
  <?php
  // Check if the form is submitted
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Get user input
  $userInput = $_POST["userInput"];

  // Validate input to ensure it's a positive integer
  if (is_numeric($userInput) && $userInput > 0 && $userInput == round($userInput)) {
  echo "<h2>Generated Numbers:</h2>";
  // Looping: foreach loop to generate even and odd numbers
  echo "<p>Even Numbers: ";
  for ($i = 0; $i <= $userInput; $i += 2) {
```

```
echo "$i ";
 }
 echo "</p>";

 echo "<p>Odd Numbers: ";
 for ($i = 1; $i <= $userInput; $i += 2) {
echo "$i ";
 }
 echo "</p>";
  } else {
 // Display an error message for invalid input
  echo "<p style='color: red;'>Please enter a valid positive integer.</p>";
  }
   }
   ?>
```

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

Output:

In this example:

1. The user is prompted to enter a number.
2. When the form is submitted, PHP checks if the input is a positive integer.
3. If the input is valid, PHP uses a foreach loop to generate and display even and odd numbers up to the user's input.
4. If the input is invalid, an error message is displayed in red.

This example showcases how PHP can be used for decision-making (checking user input) and looping (generating and displaying numbers) within an HTML document.

Practice Questions

1. Which of the following is the correct syntax for a conditional statement in PHP?

A) if condition { }
B) if (condition) { }
C) if condition:
D) if [condition] { }

2.What will the following code output?

```
$x = 10;
if ($x > 5) {
   echo "Greater";
} else {
   echo "Smaller";
}
```

A) Greater
B) Smaller
C) 10
D) Error

3. Which control structure is used to execute a block of code multiple times?

A) if
B) switch
C) for
D) echo

4. What is the purpose of the break statement in PHP?

A) To exit a function
B) To terminate a loop or switch
C) To stop the script execution
D) To continue to the next iteration

5. Which of the following is the correct syntax for a switch statement?

A) switch (expression) { case x: }
B) switch expression { case x: }
C) switch { case x: expression }
D) switch (expression) case x:

6. What will the output of the following code be?

```
$day = 3;

switch ($day) {
 case 1:
   echo "Monday";
  break;
    case 2:
   echo "Tuesday";
   break;
    case 3:
   echo "Wednesday";
   break;
    default:
   echo "Not a valid day";
}
```

A) Monday
B) Tuesday
C) Wednesday
D) Not a valid day

7. In a for loop, which part is executed only once?

A) Initialization
B) Condition
C) Increment/Decrement
D) Body of the loop

8. What does the continue statement do in a loop?

A) Exits the loop
B) Stops the current iteration and continues to the next
C) Restarts the loop
D) Skips the next iteration

9. How many times will the following code block execute?

```
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
```

A) 4 B) 5 C) 6 D) 0

10. Which of the following correctly defines a while loop?

A) while (condition) { } B) while condition { }
C) while { condition } D) while: condition { }

11. Which of the following is the correct syntax for a foreach loop?

A) foreach (array as value) { }
B) foreach array as value { }
C) foreach (array as value) { }
D) foreach (value in array) { }

12. What will the following code output?

```
$i = 0;
while ($i < 3) {
    echo $i;
    $i++;
}
```

A) 012 B) 123 C) 321 D) 00

13. What is the purpose of the else if statement?

A) To execute a block of code if the first condition is false
B) To create a loop
C) To exit from a switch statement
D) To initialize a variable

14. What will the result of the following code be?

```
$x = 10;
if ($x == 10) {
    echo "Ten";
} elseif ($x == 20) {
    echo "Twenty";
} else {
    echo "Not Ten or Twenty";
}
```

A) Ten B) Twenty
C) Not Ten or Twenty D) Error

15. Which loop will always execute at least once?

A) for B) while
C) do...while D) foreach

16. What will the following code output?

```
$day = "Saturday";
if ($day == "Saturday" || $day == "Sunday") {
    echo "Weekend";
} else {
    echo "Weekday";
}
```

A) Weekend B) Weekday
C) Saturday D) Error

17. How do you define a multi-line comment in PHP?

A) // This is a comment
B) /* This is a comment */
C) # This is a comment
D) <!-- This is a comment -->

18. What does the switch statement evaluate?

A) A single condition
B) Multiple conditions simultaneously
C) An expression
D) A function

19. In a do...while loop, when is the condition checked?

A) Before executing the loop
B) After executing the loop
C) During the loop
D) Never checked

20. What will be the output of the following code?

```
$x = 5;
if ($x < 10) {
    echo "Less than 10";
} else {
    echo "10 or more";
}
```

A) Less than 10 B) 10 or more
C) 5 D) Error

Ans : 1-b,2-a,3-c,4-b,5-a,6-c,7-a,8-b,9-b,10-a,11-a&c,12-a,13-a,14-a,15-c,16-a,17-b,18-c,19-

"The secret to getting ahead is getting started." – Mark Twain 🚀

Professor Academy

PG TRB

COMPUTER SCIENCE

COURSE DETAILS

+91 707070 1005
+91 707070 1009

www.professoracademy.com

Professor Academy

PG TRB 2025 ONLINE COURSE

New Syllabus 2024

Complete coverage of **New Syllabus 2024**

Online Live Classes

Online Live Classes
200+ hrs. of Lectures

100+ Test Series

- Daily test
- Unit wise test
- Full-length test

REC

Recorded Access

Recorded Access
24/7 Availability

Study Material

Study Material
12 Printed Books + Class notes

Support

Technical & Academic Support

www.professoracademy.com

Professor Academy

VIRTUAL CLASS FEATURES

Start Date: December 06, 2024

Duration of Classes: 5 months (up to exam date)

Class Schedule:

- **Morning Session:** 5:00 AM - 7:00 AM
- **Evening Session:** 7:30 PM - 9:00 PM

(subject to change based on exam announcements)

Key Features:

- Interactive online live classes
- Weekly schedule updates via WhatsApp community
- Sessions conducted through Zoom for ease of access

Professor Academy

COURSE BENEFITS

APP FEATURES

- Login ID and password will be provided to the candidates to access the mobile app after enrolling in course.
- **Access to Missed Live Sessions** as Recorded Videos.
- Validity for Recorded Sessions and Test Series will be provided upto 1 year in our mobile app (Upto date of the examination)
- **Access to Missed Live Sessions as Recorded Videos.**
- **Mobile App** for learning (For Android users)
- **Website** for easy access from any device

Professor Academy

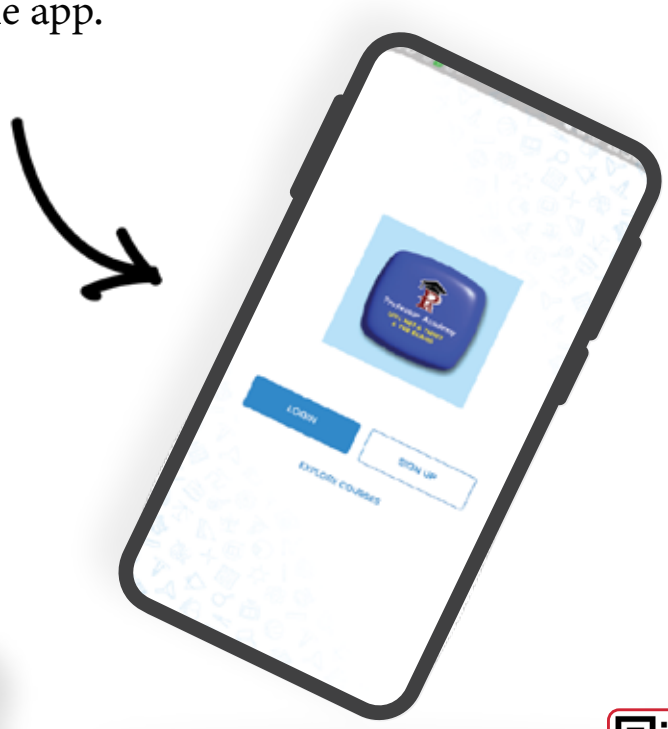
TEST SERIES

Extensive Test Series to Boost Your Preparation!

Get access to 100+ Test Series, including:

- **Daily Tests** to reinforce learning every day
- **Unit-Wise Tests** for focused revision
- **Full-Length Tests** to simulate the real exam experience

Comprehensive Study Materials as soft copy PDFs, available in our mobile app.



Download & Explore!

www.professoracademy.com

Professor Academy

STUDY MATERIALS

Enroll today and receive 12 Printed Books right at your doorstep

- **10 Subject-Based Books** covering all major topics
- **1 Tamil Eligibility Book**
- **1 Previous Year Question Bank**

(Subject Book + PYQ Bank available only in English Medium)

Comprehensive Study Materials as soft copy PDFs, available in our mobile app.

PG TRB COMPUTER SCIENCE

Based on New Syllabus

Rs. 300 for courier charge payments is excluded

www.professoracademy.com

Professor Academy

PAYMENT DETAILS& ENROLLMENT PROCESS

BOOKS ONLY	COURSE ONLY	COURSE + BOOKS
₹3999	₹8999	₹11999
ACTUAL FEE ~~₹5999~~	ACTUAL FEE ~~₹14999~~	ACTUAL FEE ~~₹19999~~

An additional charge of Rs. 300 for courier charge payments is excluded in the original course fee.

Step 1: Register Online on Our Website or App.

SCAN HERE
TO REGISTER

Step 2: Choose Your Course and Complete Payment

Step 3: Receive Login Details and Begin Your Journey!

Step 4: Our dedicated support team is here to assist you at every step.

NOTE* Books will be provided only after the full payment of fees has been made.

www.professoracademy.com

Professor Academy

Professor Academy's Pride
TRB State Rankers

ISWARYA P

DEVASAGAYAM D

RAJESHWARI N

Achiever's Testimonials

Line by line teaching helps me to take good notes. They teach me lessons thoroughly, and then only I attend their tests. Their test makes me more focused on preparation.

VINITHA. M | STATE 6TH RANK

Professor Academy's test series are good. I write each and every test like TRB exam. I am very happy to attend Professor Academy's test. Happy to see my test result. In a similar way, I secured state rank on PG TRB exam (just assume exam like Professor Academy's test).

AMUTHA VIJAYALAKSHMI. A | STATE 7TH RANK

Professor Academy's motivation and their notes are very helpful for me. Their crystal-clear explaining faculties are the secret of my success. Their friendly guidance boosts my exam preparation.

HEMA. P | STATE 8TH RANK

The reason for my success is Professor Academy. If not for Professor Academy, I wouldn't have achieved this. The recorded classes are very useful for me. It made studying easier and gave me confidence. Therefore, I recommend Professor Academy to everyone who wants to succeed in their studies.

ARUNA RAJESWARI. R | STATE 10TH RANK

www.professoracademy.com

Click here to Watch more Achiever's Talks