1...

Introduction to PHP

Learning Objectives...

- To understand the concept of HTTP, Web Server and Web Browser.
- To learn in detail about PHP.
- To study the Language Basics.

1.1 INTRODUCTION

- The World Wide Web is a vast network of interconnected documents and resources accessed via the internet. It relies on three core components: HTTP (Hypertext Transfer Protocol), Web Servers, and Web Browsers

1.1.1 HTTP Basics

Definition:

- HTTP (Hypertext Transfer Protocol) is the foundation of data exchange on the web. It's a request-response protocol that governs how browsers and servers communicate.

Functionality:

- A browser sends an HTTP request to a server, asking for a specific resource (like a web page). The server then sends back an HTTP response, containing the requested resource or an error message.

1.1.2 Web Server

- **Definition:** A web server is a computer program or system that delivers web pages and other content to users' web browsers.
- **Functionality:** It stores website files (HTML, images, CSS, etc.) and responds to requests from browsers. When a user requests a page, the server retrieves the relevant files and sends them back to the browser.
- **Examples:** Popular web servers include Apache, Nginx, and Microsoft IIS. In PHP we have to use Xampp Server.

1.1.3 Web Browser

Definition:

- A web browser is a software application used to access and display web pages and other content on the internet.

Functionality:

- It sends HTTP requests to web servers, receives the responses, and renders the content for the user to view.

1.2 INTRODUCTION TO PHP

What is PHP?

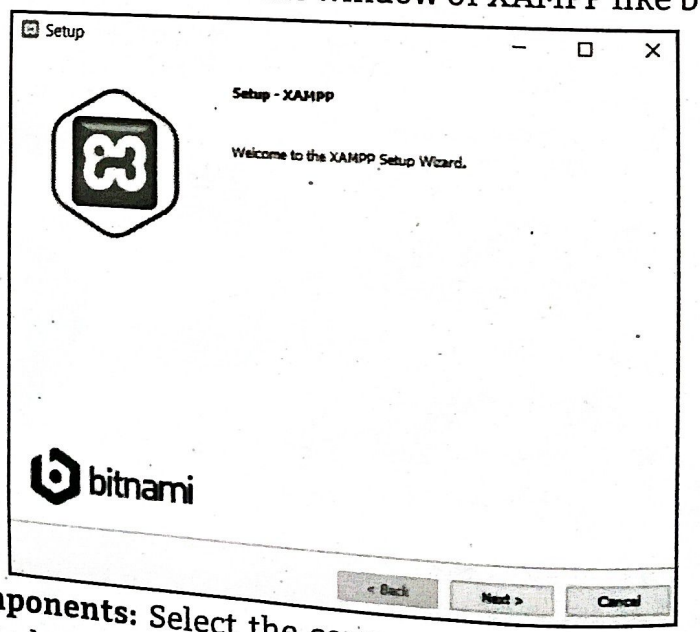
- PHP (Hypertext PreProcessor) is a general-purpose programming language developed by Rasmus Lerdorf in 1994 for web development. This is one of the most popular programming languages for beginners in web development because of its simplicity, large community and accessibility.

Steps to setup PHP environment on a local machine:

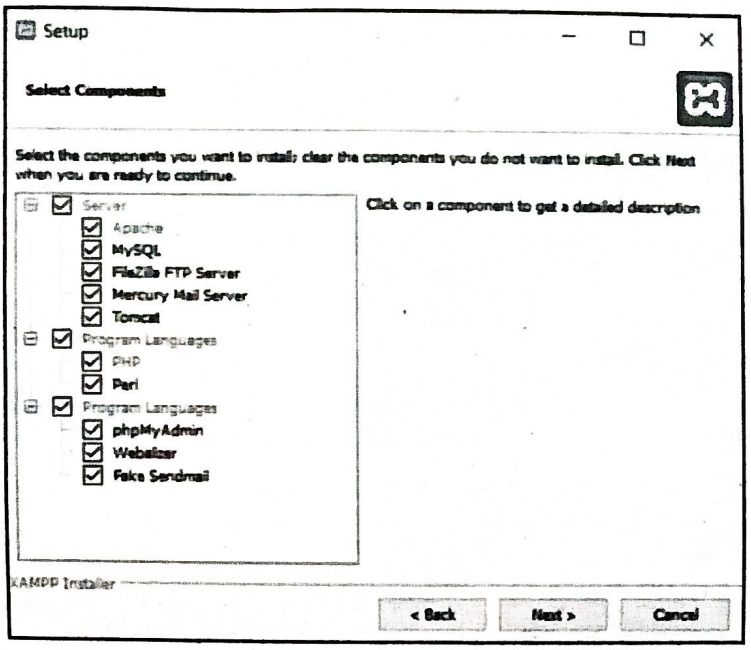
- There are basically two ways to set up PHP on a local machine which are:
  - Using all in one package (XAMPP & WAMP). (recommended)
  - Manually install all the required packages (MySQL, PHP & Apache) and configure them.
- We are going to cover an easy and almost error-free method to install PHP on a local machine which is by using all in one package called XAMPP.

1.2.1 Steps to Installing XAMPP

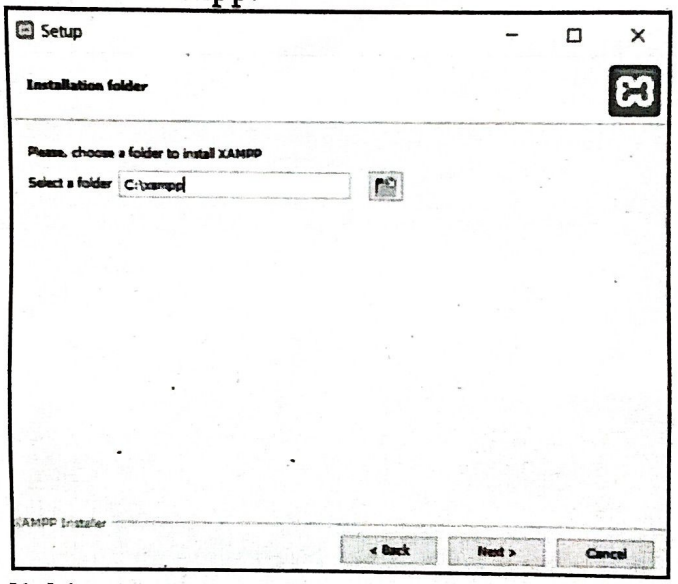
1. **Downloading XAMPP:** You can download the XAMPP software from the official website here with the latest windows version and latest PHP version.
2. **Open the downloaded .exe file:** After opening the downloaded file you will see a popup from windows, click yes and proceed further.
3. **Click on 'next':** You'll see a welcome window of XAMPP like below, click next.



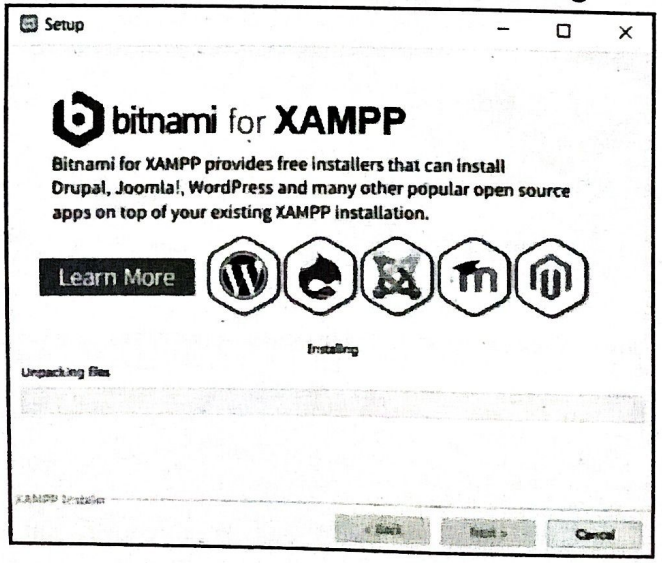
4. **Select the components:** Select the components you want to install. Please select the MySQL and phpMyAdmin components, all other components are optional for this tutorial.



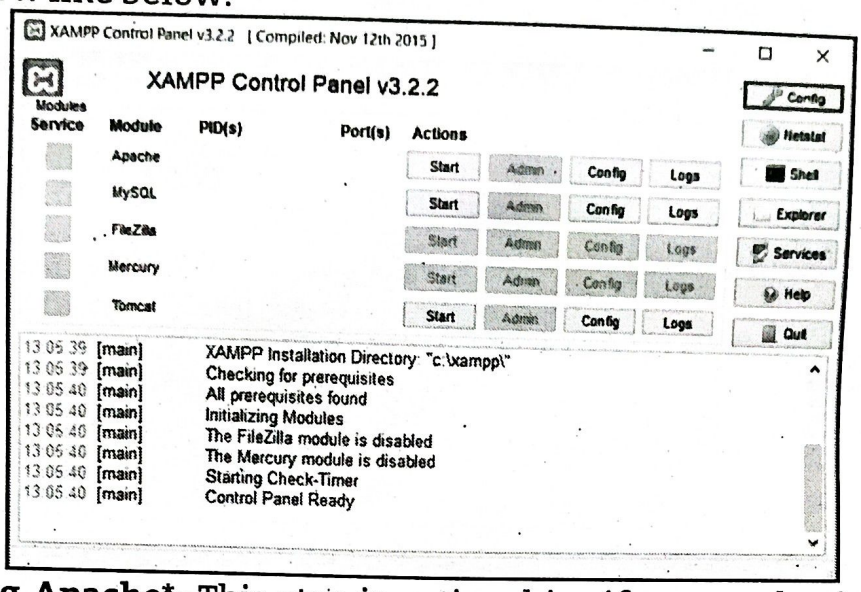
5. **Select the installation location:** Select the location you want to install the XAMPP, the default is C:\xampp.



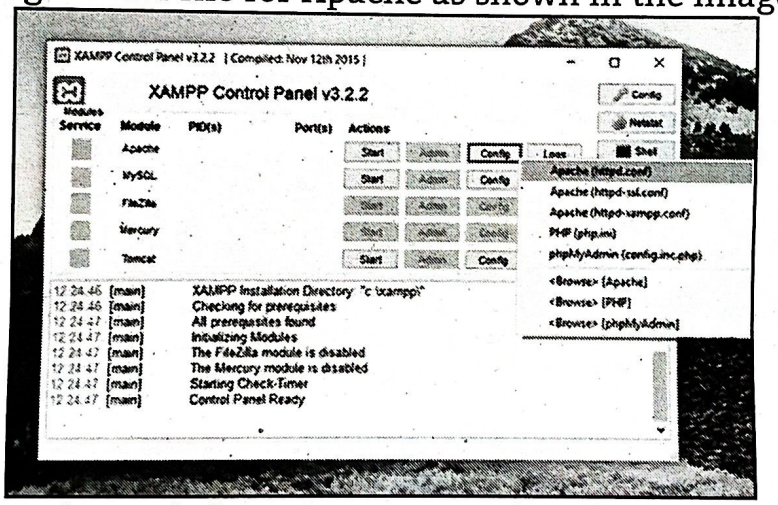
6. **Click next:** You clicking next, your installation will begin.



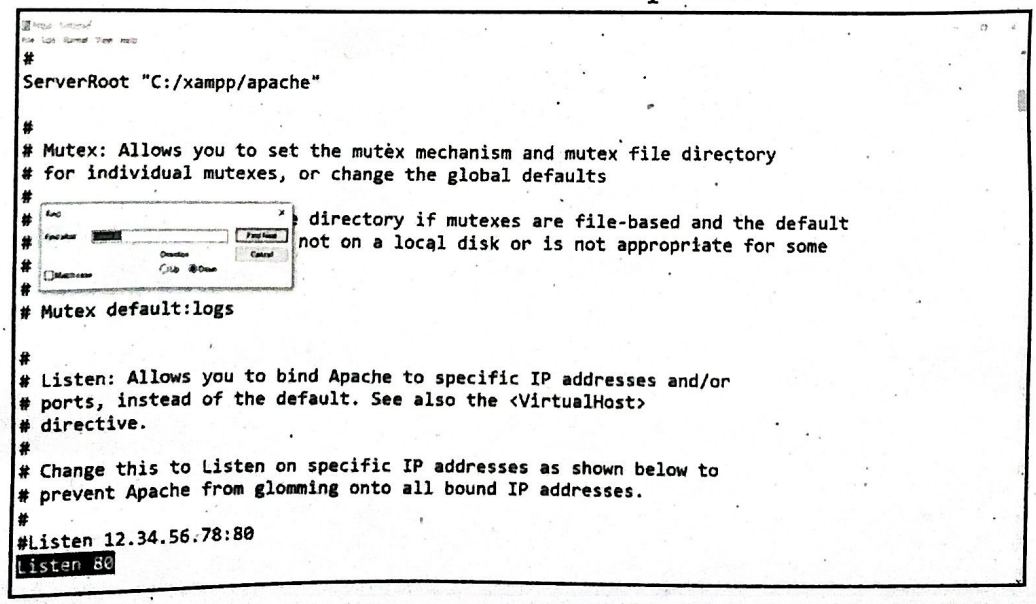
7. **Open XAMPP control panel:** After successfully installing the XAMPP in your local machine open the control panel by searching in the windows search bar for 'XAMPP control panel' or by going to the installation directory of XAMPP. You'll see a window like below.



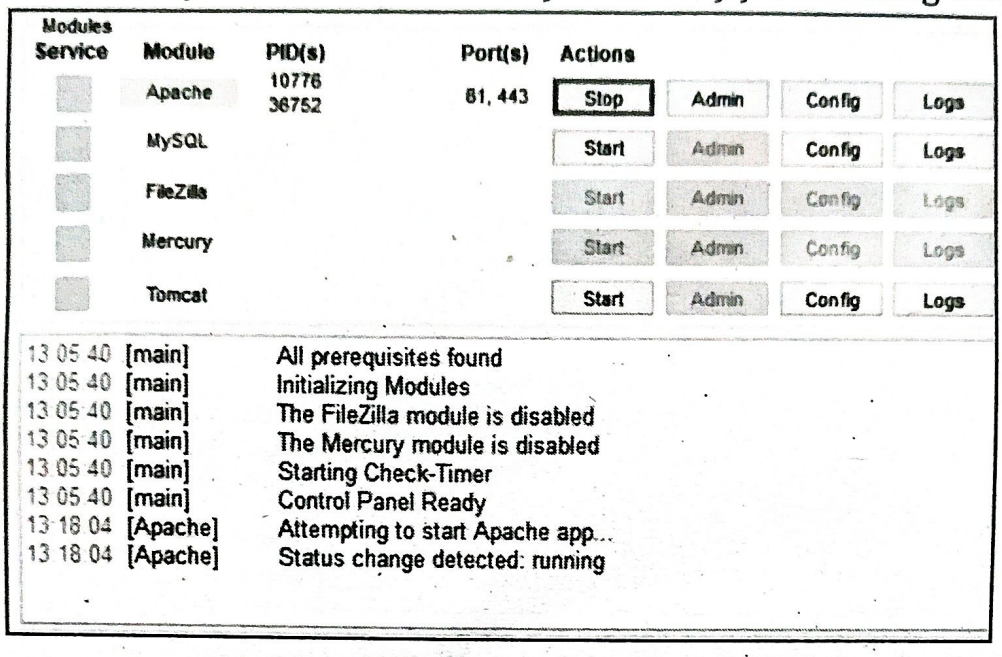
8. **Configuring Apache*:** This step is optional i.e. if you are having some issues on windows 10 related to blocked port.
Open the configuration file for Apache as shown in the image below.



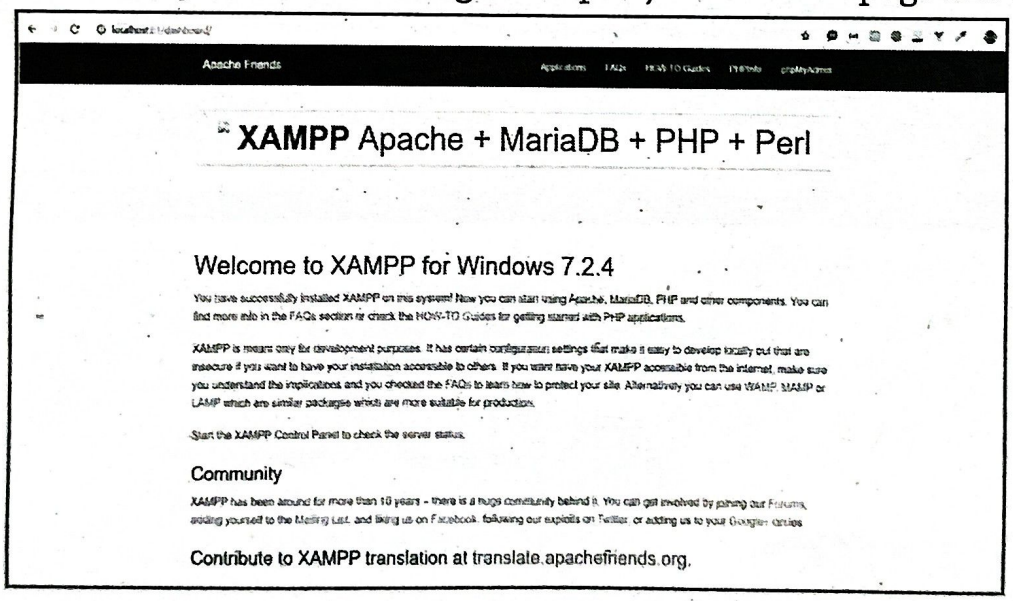
Press CTRL + F and search for LISTEN 80. Replace the port 80 with something like 8fi and save the file and restart the control panel.



9. **Start the Apache server:** Start the Apache server by clicking the start button you'll see a port number just in front of the Apache column. You can stop the service whenever you want and start any service by just clicking the start button.



10. **Checking the installation:** Go to your browser and type localhost:8fi (or simply localhost in case you haven't changed the port). You'll see a page like below.

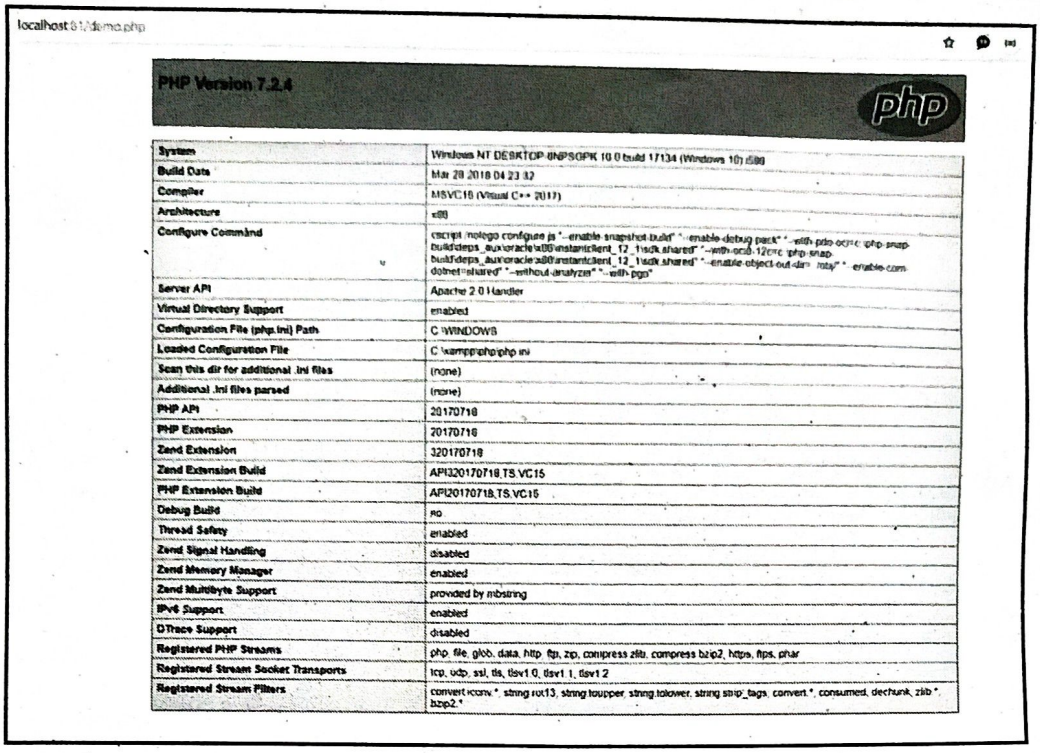


11. **Checking PHP installation:** Create a php file in the htdocs folder in your installation directory (C:/XAMPP/htdocs) and add the following code in it.

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

12. Save the file as demo.php and go to your browser and type localhost:81/demo.php (or simply localhost/demo.php in case you haven't changed the port).

If the installation goes fine you'll see a result like below



- If all the above things have gone fine then Congratulations! You have successfully set up a PHP development environment in your local machine.
- **Examples:** Popular web browsers include Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

1.2.2 Setting up a Development Environment in PHP

- To start developing PHP applications, you need a proper development environment that includes:
  - PHP Interpreter
  - Web Server (Apache or Nginx)
  - Database (like MySQL or MariaDB)
  - Text Editor/IDE
  - (Optionally) phpMyAdmin for managing databases
- You can set this up manually or by using all-in-one packages like XAMPP, WAMP, MAMP, or Laragon.

Using XAMPP (Recommended for Beginners)

What is XAMPP?

- XAMPP is a free, cross-platform package that includes:
  - Apache (Web Server)
  - MySQL (Database)
  - PHP (Interpreter)
  - phpMyAdmin (Database management tool)

1.3 LANGUAGE BASICS

1.3.1 Operators

- In PHP, operators are special symbols used to perform operations on variables and values. Operators help you perform a variety of tasks, such as mathematical calculations, string manipulations, logical comparisons, and more. Understanding operators is essential for writing effective and efficient PHP code.
- Operators are used to perform operations on variables and values.

(i) Arithmetic Operators

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

Operator	Name	Description
+	Addition	$x + $y returns the sum of $x and $y.
–	Subtraction	$x – $y returns the difference of $x and $y.
*	Multiplication	$x * $y returns the product of $x and $y.
/	Division	$x / $y returns the quotient of $x and $y.
%	Modulo	$x % $y returns the integer remainder of $x divided by $y.
**	Exponentiation	$x ** $y returns $x raised to the power $y.

```
<?php
  $x = 0;
  $x = $x + 4; // $x is now 4
  $x = $x - 1; // $x is now 3
  $x = $x * 8; // $x is now 24
  $x = $x / 4; // $x is now 6
  $x = $x % 4; // $x is now 2
  $x = $x ** 3; // $x is now 8
?>
```

- Operators are used to perform operations on variables and values.
- The division operator returns a float unless its operands are integers and the result is a whole number, in which case it returns an integer.
- Operands of modulo are converted to integers before calculating. The modulo result will have the same sign as the dividend.

(ii) Assignment Operators:

- The basic assignment operator is =, which takes the right-hand operand and assigns it to the variable that is the left-hand operand. PHP also has a number of additional assignment operators that are shortcuts for longer expressions.

Arithmetic Assignment Operators:

- There is an assignment operator for each arithmetic operator above.

Operator	Name
+=	Addition Assignment
-=	Subtraction Assignment
*=	Multiplication Assignment
/=	Division Assignment
%=	Modulo Assignment
**=	Exponentiation Assignment

```
<?php
  $x = 0;
  $x += 4; // $x is now 4
  $x -= 1; // $x is now 3
  $x *= 8; // $x is now 24
  $x /= 4; // $x is now 6
  $x %= 4; // $x is now 2
  $x **= 3; // $x is now 8
?>
```

- Assignment operators in PHP to assign values to variables. Assignment operators are shorthand notations to perform arithmetic or other operations while assigning a value to a variable. For instance, the "=" operator assigns the value on the right-hand side to the variable on the left-hand side.
- Additionally, there are compound assignment operators like +=, -= , *=, /=, and %= which combine arithmetic operations with assignment. For example, "$x += 5" is a shorthand for "$x = $x + 5", incrementing the value of $x by 5. Assignment operators offer a concise way to update variables based on their current values.

Assignment Operators List:

- The following table highlights the assignment operators that are supported by PHP:

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A

Operator	Description	Example
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand..	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Program 1.1:

```
<?php
    $a = 42;
    $b = 20;
    $c = $a + $b;
    echo "Addition Operation Result: $c \n";
    $c += $a;
    echo "Add AND Assignment Operation Result: $c \n";
    $c -= $a;
    echo "Subtract AND Assignment Operation Result: $c \n";
    $c *= $a;
    echo "Multiply AND Assignment Operation Result: $c \n";
    $c /= $a;
    echo "Division AND Assignment Operation Result: $c \n";
    $c %= $a;
    echo "Modulus AND Assignment Operation Result: $c";
?>
```

Output:

Addition Operation Result: 62.
Add AND Assignment Operation Result: 104
Subtract AND Assignment Operation Result: 62
Multiply AND Assignment Operation Result: 2604
Division AND Assignment Operation Result: 62
Modulus AND Assignment Operation Result: 20

(iii) Comparison operators:

- Comparison operators are used to compare two values and determine their relationship. These operators return a Boolean value, either True or False, based on the result of the comparison.
- The following table highlights the comparison operators that are supported by PHP. Assume variable $a holds 10 and variable $b holds 20, then,

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then the condition becomes true.	($a == $b) is not true
!=	Checks if the values of two operands are equal or not, if values are not equal then the condition becomes true.	($a != $b) is true
>	Checks if the value of the left operand is greater than the value of the right operand, if yes then the condition becomes true.	($a > $b) is false
<	Checks if the value of the left operand is less than the value of the right operand, if yes then the condition becomes true.	($a < $b) is true
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	($a >= $b) is false
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	($a <= $b) is true

(iv) Bitwise Assignment Operators:

- PHP also has bitwise assignment operators, which let you perform bitwise operations and store the result in the same variable. Please refer to the below table for the list of bitwise assignment operators.

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
&=	Bitwise AND assignment	$a &= $b; // $a = $a & $b
|=	Bitwise OR assignment	$a |= $b; // $a = $a | $b

Operator	Description	Example
^=	Bitwise XOR assignment	`$a ^= $b; // $a = $a ^ $b`
<<=	Left shift AND assignment	`$a <<= 2; // $a = $a << 2`
>>=	Right shift AND assignment	`$a >>= 2; // $a = $a >> 2`

(v) Logical operators:

- Logical operators are used to combine conditional statements. These operators allow you to create more complex conditions by combining multiple conditions together.
- Logical operators are generally used in conditional statements such as if, while, and for loops to control the flow of program execution based on specific conditions.
- The following table highlights the logical operators that are supported by PHP. Assume variable $a holds 10 and variable $b holds 20, then,

Operator	Description	Example
and	It is called Logical AND operator. If both the operands are true then the condition becomes true.	(A and B) is true
or	It is called Logical OR operator. If any of the two operands are non zero then the condition becomes true.	(A or B) is true
&&	It is called Logical AND operator. The AND operator returns true if both the left and right operands are true.	(A && B) is true
||	It is called Logical OR operator. If any of the two operands are non zero then the condition becomes true.	(A || B) is true
!	It is called Logical NOT operator. Used to reverse the logical state of its operand. If a condition is true then Logic becomes false.	!(A)

Program 1.2:

```
<?php
   $a = 42;
   $b = 0;
   if ($a && $b)
```

```
{
   echo "Result1 : Both a and b are true \n";
} else {
   echo "Result1 : Either a or b is false \n";
}
if ($a and $b)
{
   echo "Result2 : Both a and b are true \n";
} else {
   echo "Result2 : Either a or b is false \n";
}
if ($a || $b)
{
   echo "Result3 : Either a or b is true \n";
} else {
   echo "Result3 : Both a and b are false \n";
}
if ($a or $b)
{
   echo "Result4 : Either a or b is true \n";
} else {
   echo "Result4 : Both a and b are false \n";
}
$a = 10;
$b = 20;
if ($a)
{
   echo "Result5 : a is true \n";
} else {
   echo "Result5 : a is false \n";
}
if ($b)
{
   echo "Result6 : b is true \n";
```

```
    } else {
        echo "Result6 : b is false \n";
    }
    if (!$a)
    {
        echo "Result7 : a is true \n";
    } else {
        echo "Result7 : a is false \n";
    }
    if (!$b)
    {
        echo "Result8 : b is true \n";
    } else {
        echo "Result8 : b is false";
    }
?>
```

Output:

Result1 : Either a or b is false
Result2 : Either a or b is false
Result3 : Either a or b is true
Result4 : Either a or b is true
Result5 : a is true
Result6 : b is true
Result7 : a is false
Result8 : b is false

1.3.2 Flow-Control Statements in PHP

- Flow-control statements determine the execution path of a PHP script based on conditions or loops. They help control how and when code executes.

1.3.2.1 Conditional Statements

- These allow the program to make decisions based on conditions.

1. if Statement:

- The if statement is used when you want to execute one (or more) statements only if a condition is true.
- Executes a block of code if the condition is true.

Syntax:

```
if (condition) {
// Code to execute if condition is true
}
```

Program 1.3:

```
<?php
    $age = 20;
    if ($age >= 18)
    {
        echo "You are eligible to vote.";
    }
?>
```

Output:

You are eligible to vote.

2. if...else Statement:

- The if...else statement is used when you want to execute one block of code if a condition is true, and another block of code if the condition is false.

Syntax:

```
if (condition) {
// Code if condition is true
} else {
// Code if condition is false
}
```

Program 1.4:

```
<?php
    $marks = 40;
    if ($marks >= 50)
    {
        echo "Pass";
    } else
    {
        echo "Fail";
    }
?>
```

Output:

Fail

3. if...elseif...else

- The if...elseif...else statement allows you to test multiple conditions in sequence. It executes the first block of code where the condition is true. If none of the conditions are true, the else block is executed.

Syntax:

```
if (condition1) {
// Code block 1
} elseif (condition2) {
// Code block 2
} else {
// Code block if none of the above conditions are true
}
```

Program 1.5:

```
<?php
   $grade = 85;
   if ($grade >= 90)
   {
      echo "A+";
   } elseif ($grade >= 75)
   {
      echo "A";
   } elseif ($grade >= 60)
   {
      echo "B";
   } else
   {
      echo "Fail";
   }
?>
```

Output:

```
A
```

4. switch Statement:

- The switch statement is used to perform different actions based on the value of a single variable or expression. It is an alternative to using multiple if...elseif...else statements when comparing the same variable to different values.

- Switch compares the given expression with each case. break prevents fall-through (executes only the matching case). default is optional and executes when no case matches.
- It improves readability over multiple if...elseif statements when checking a single variable.

Program 1.6:

```
<?php
   $day = "Friday"; switch ($day)
{
    case "Monday":
    echo "Start of the week"; break;
    case "Tuesday":
    echo "Second day"; Break;
    case "Friday":
    echo "Second last day"; break;
    default:
    echo "Another day";
 }
 ?>
```

Output:

Second last day

1.3.2.2 Looping Statements

- A loop is a control structure that allows you to repeatedly execute a block of code as long as a specified condition is true.
- Reducing code repetition.
- Automating repetitive tasks like printing numbers, processing arrays, etc.
- Controlling the flow based on dynamic conditions.

1. while Loop:

- The whileloop in PHP is used to execute a block of code repeatedly as long as a given condition is true. The condition is evaluated before the code block is executed.

Syntax:

```
while (condition) {
// Code to be executed
}
```

Program 1.7:

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

Output:

1
2
3
4
5

2. do...while Loop:

- The do...while loop in PHP is similar to the while loop, but it executes the code block at least once before checking the condition at the end of the loop.

Syntax:

```
do {
// Code to be executed
} while (condition);
```

Program 1.8:

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

Output:

1
2
3
4
5
6
7
8

3. for Loop

- The forloop in PHP is used to execute a block of code a specific number of times. It is best suited when the number of iterations is known in advance.

Syntax of for loop:

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

Program 1.9:

```
<?php
    for ($i = 1; $i <= 5; $i++)
    {
        echo "Number: $i <br>";
    }
?>
```

Output:

Number: 1
Number: 2
Number: 3
Number: 4
Number: 5

4. foreach Loop:

- The foreach loop in PHP is used to iterate over arrays. It simplifies accessing each element in an array without using an index.

Syntax:

```
foreach ($array as $value) {
// Code to use $value
}
```

Program 1.10:

```
<?php
    $stud = ["Umar", "Ali", "Ammar", "Osman"];
    foreach ($stud as $stud)
    {
        echo $stud . "<br>";
    }
```

Output:

Umar
Ali
Ammar
Osman

Program 1.11: Program for associative arrays

```
<?php
    $marks = ["php" => 70, "python" => 75];
    foreach ($marks as $subject => $mark)
    {
        echo "$subject: $mark <br>";
    }
?>
```

Output:

php: 70
python: 75

1.3.2.3 Jump Statements

1. **break:** Exits the loop or switch.

Program 1.12:

```
<?php
   for ($i = 1; $i <= 10; $i++)
   {
      if ($i == 5) break;
      echo $i . " ";
   }
?>
```

Output:

```
1 2 3 4
```

2. **continue:** Skips current iteration, moves to next.

Program 1.13:

```
<?php
   for ($i = 1; $i <= 5; $i++)
   {
      if ($i == 3) continue; echo $i . " ";
   }
?>
```

Output:

```
1 2 4 5
```

1.3.3 Including Code Embedding PHP in Web Pages

- PHP (Hypertext Preprocessor) is a server-side scripting language that can be embedded directly into HTML to add dynamic content to web pages.
- We can use PHP in HTML code by simply adding a PHP tag without doing any extra work.

Program 1.14: First open the file in any editor and then write HTML code according to requirement. If we have to add PHP code then we can add by simply adding <?php ..... ?> tags in between and add your PHP code accordingly.

```
<!DOCTYPE html>
<html>
<head>
<title>PHP Embedded in HTML</title>
```

```
</head>
<body>
<h1>Welcome to SYBSc(computer Application)</h1>
<?php
// PHP block inside HTML
   $name = "Maseera";
   $name1 = "Aliza";
   $class = "SYBSc(CA)";   // put in quotes and add semicolon
   echo "Name: $name <br>";
   echo "Name: $name1 <br>";
   echo "Class: $class <br>";
?>
</body>
</html>
```

Output:

Maseera

SYBSc(CA)

PHP include Statements:

- It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.
- The include statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
- Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

Program 1.15: Assume we have a file called "student.php", with some variables defined:

```
<?php
   $rollno=01;
   $name='Ali';
   $rollno=02;
   $name='Ammar';
?>
```

- Then, if we include the "student.php" file, the variables can be used in the calling file:

```
<html>
<body>
<h1>Welcome to my home page!</h1>
```

```
<?php
    include 'student.php';
    echo "We have student details: Roll No = $rollno, Name = $name";
?>
</body>
</html>
```

Output:

Welcome to my home page!

We have student details: Roll No = 2, Name = Ammar

require in PHP:

- The require statement is also used to include a file.
- If the specified file cannot be found or there is an error during inclusion, require generates a fatal error (E_COMPILE_ERROR), and the script terminates execution.
- This is suitable for critical files (e.g., database connection files, configuration files) where the application cannot function correctly without them.

Program 1.16: Program of require

```
<?php
// config.php
    $db_host = "localhost";
    $db_user = "root";
?>
<?php
// index.php
    require 'config.php';
    echo "Database host: " . $db_host;
    require 'another_non_existent_file.php'; // This will generate a fatal error and stop the script
    echo "<p>This line will NOT be executed.</p>";
?>
```

Output:

Database host: localhost

Warning: include(another_non_existent_file.php): failed to open stream: No such file or directory...

This line WILL be executed with include.

Sample program

Program 1 .17: Check Even or Odd

```
<?php
$num = 7;

if ($num % 2 == 0) {
    echo "$num is Even";
} else {
    echo "$num is Odd";
}
?>
```

Output:

7 is Odd

Program 1.18: Print multiplication table using loop.

```
<?php
$num = 5;
for ($i = 1; $i <= 10; $i++)
{
    echo "$num x $i = " . ($num * $i) . "<br>";
}
?>
```

Output:

5 × 1 = 5
5 × 2 = 10
5 × 3 = 15
5 × 4 = 20
5 × 5 = 25
5 × 6 = 30
5 × 7 = 35
5 × 8 = 40
5 × 9 = 45
5 × 10 = 50

Program 1.19: Find Largest of Three Numbers

```
<?php
  $a = 10;
  $b = 25;
```

```
$c = 15;

if ($a >= $b && $a >= $c)
{
   echo "$a is the largest";
} elseif ($b >= $a && $b >= $c)
{
   echo "$b is the largest";
} else
{
   echo "$c is the largest";
}
?>
```

Output:

25 is the largest.

Check Your Understanding

1. PHP is a .....
   (a) Client-side scripting language (b) Server-side scripting language
   (c) Database language (d) Markup language
2. Tag is used to embed PHP code in HTML?
   (a) <php> (b) <?php ... ?>
   (c) <script> (d) <?HTML?>
3. ..... of the following is a web browser?
   (a) Apache (b) XAMPP
   (c) Chrome (d) MySQL
4. ..... is the default port for HTTP.
   (a) 20 (b) 21 (c) 80 (d) 8080
5. ..... function is used to output text in PHP.
   (a) write() (b) echo()
   (c) printline() (d) display()

Answers

1. (b)	2. (b)	3. (c)	4. (c)	5. (b)

Practice Questions

Q.I Answer the following questions in short:

1. What does HTTP stand for?
2. Define a web browser and name two examples.
3. Mention two popular web servers used with PHP.
4. What is the purpose of PHP in web development?
5. Name any two flow control statements in PHP.
6. What is the use of echo in PHP?
7. Write a simple PHP code to print "Hello World".
8. What is the difference between == and === in PHP?
9. Define a PHP operator with an example.

Q.II Answer the following questions in detail:

1. Explain the HTTP request-response cycle in detail.
2. What is a Web Server? Describe its role in delivering web content with examples.
3. Compare and contrast Web Server and Web Browser with examples.
4. Explain the steps to install PHP on a local machine using XAMPP or WAMP.
5. Explain different types of operators available in PHP with examples.
6. Discuss various flow-control statements in PHP with suitable examples.
7. Explain embedding PHP code within HTML. Give a complete example.
8. Describe the process of handling an HTTP GET and POST request using PHP.
9. Discuss the significance and types of flow control in PHP programming. Provide syntax and examples.

***