

CHAPTER 1

Getting Started with PHP

Introduction to PHP

PHP is a server-side and general-purpose scripting language that is especially suited for web development. It stands for **Hypertext Preprocessor**. It's a recursive acronym because the first word itself is also an acronym.

PHP was created by Rasmus Lerdorf in 1994. It's currently maintained by the PHP Development Team.

PHP is a server-side language

When you open a website on your web browser, for example, <https://www.phptutorial.net>. The web browser sends an HTTP request to a web server where phptutorial.net is located. The web server receives the request and responds with an HTML document.

In this example, the web browser is a client while the web server is the server. The client requests for a page, and the server serves the request.

PHP runs on the web server, processes the request, and returns the HTML document.

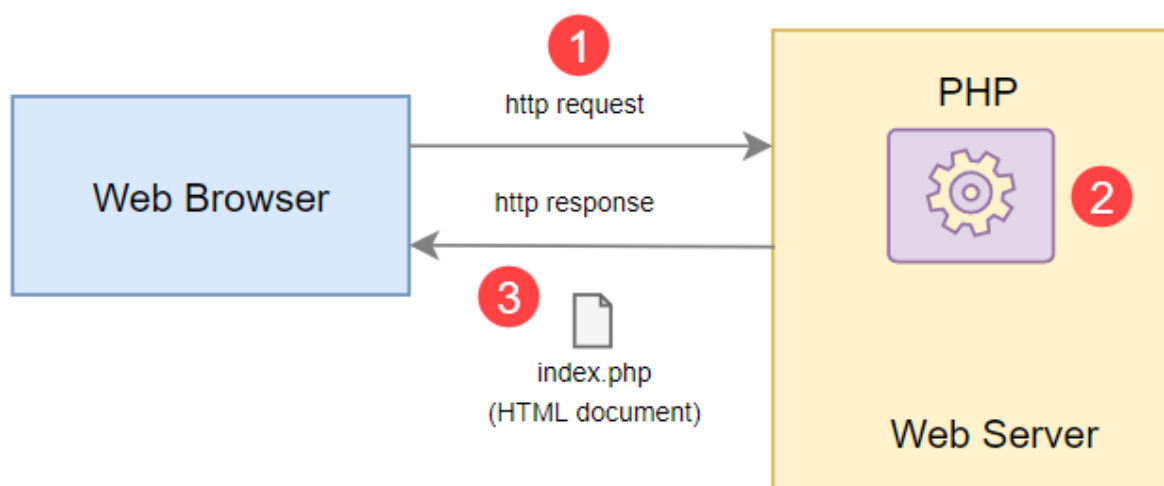


Figure 1: The illustration how PHP works

Tools needed for PHP Coding

There are 4 three tools that need to be prepared to start coding PHP:

1. **PHP interpreter** – a program to run PHP on the server
2. **Text Editor** – for writing PHP code
3. **Web Server** – service for running PHP
4. **Web Browser** – to view or open websites

Therefore, it's easier to find an all-in-one software package that includes PHP, a web server, and a database server. One of the most popular PHP development environments is **XAMPP**.

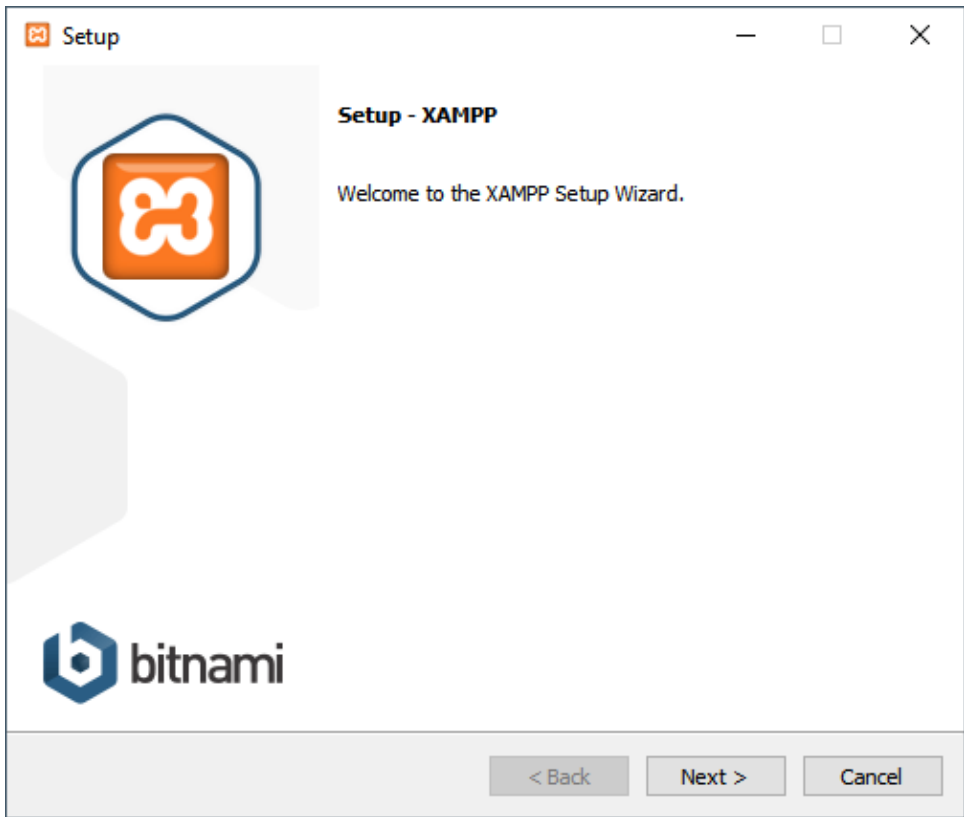
Download and Install XAMPP

To install XAMPP on windows, you can go to the XAMPP official website and download the suitable version for your platform. We'll use XAMPP ver. 7.2.2

To install XAMPP on Windows, you can follow these steps:

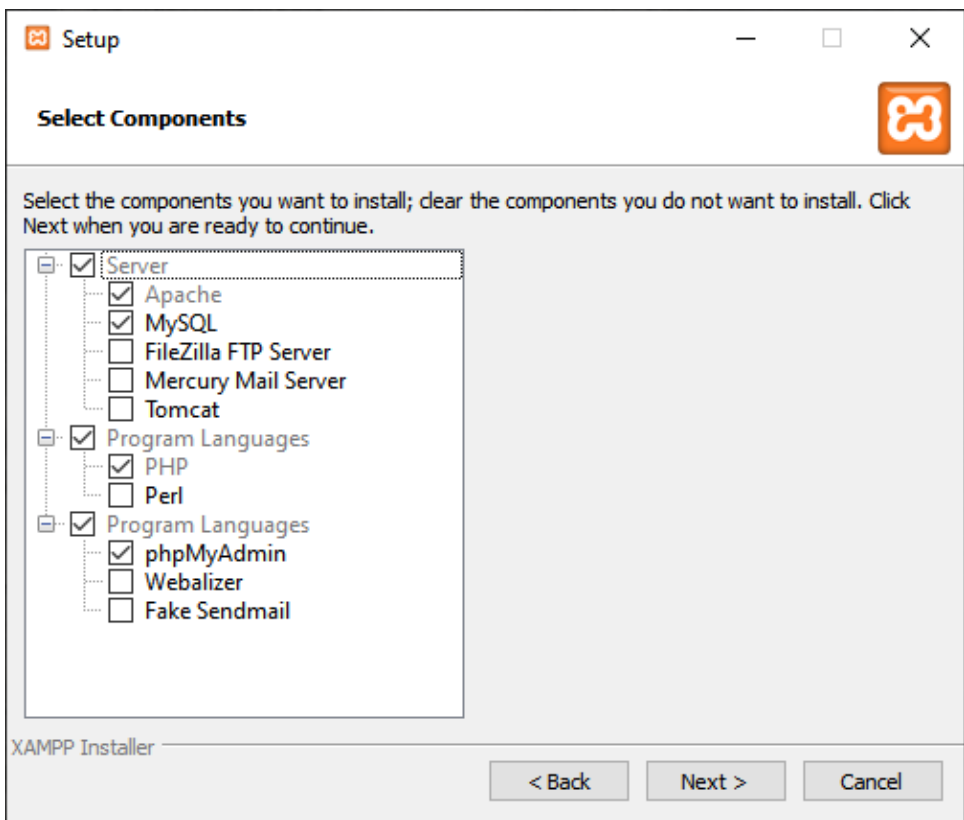
Step 1. Start the installation

Double-click the downloaded file to start setting up XAMPP:



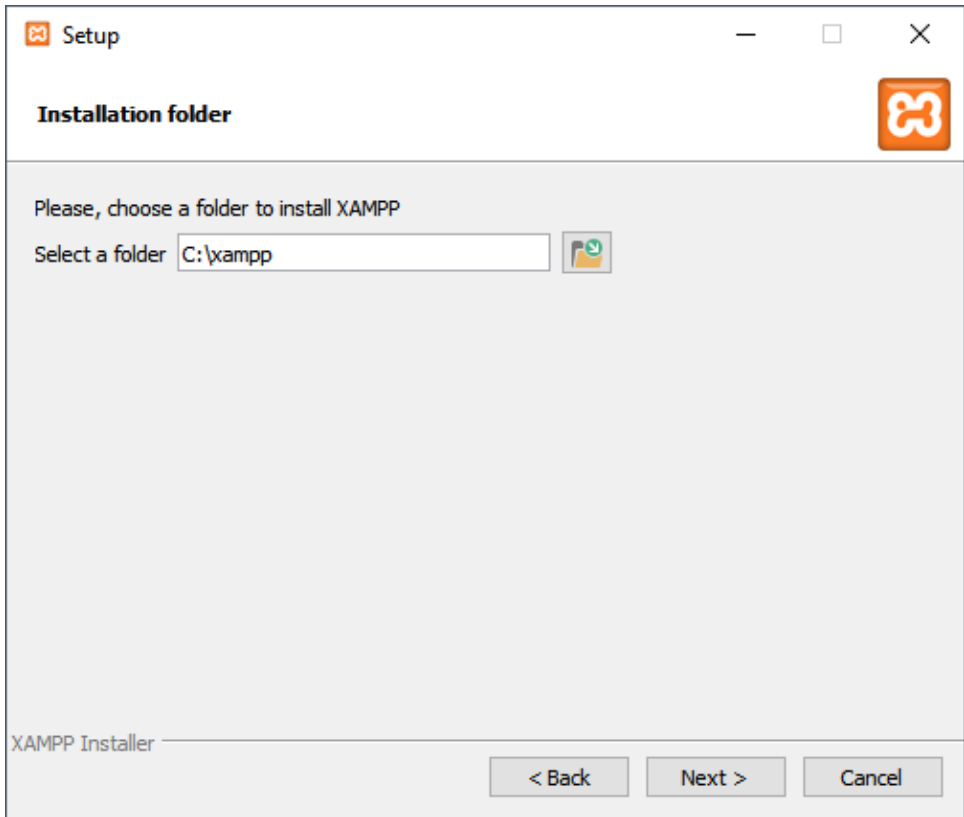
Step 2. Select components to install

Select the components that you want to install. In this step, you can select Apache, MySQL, PHP, and phpMyAdmin, deselect other components like the following, and click the Next button to go to the next step.



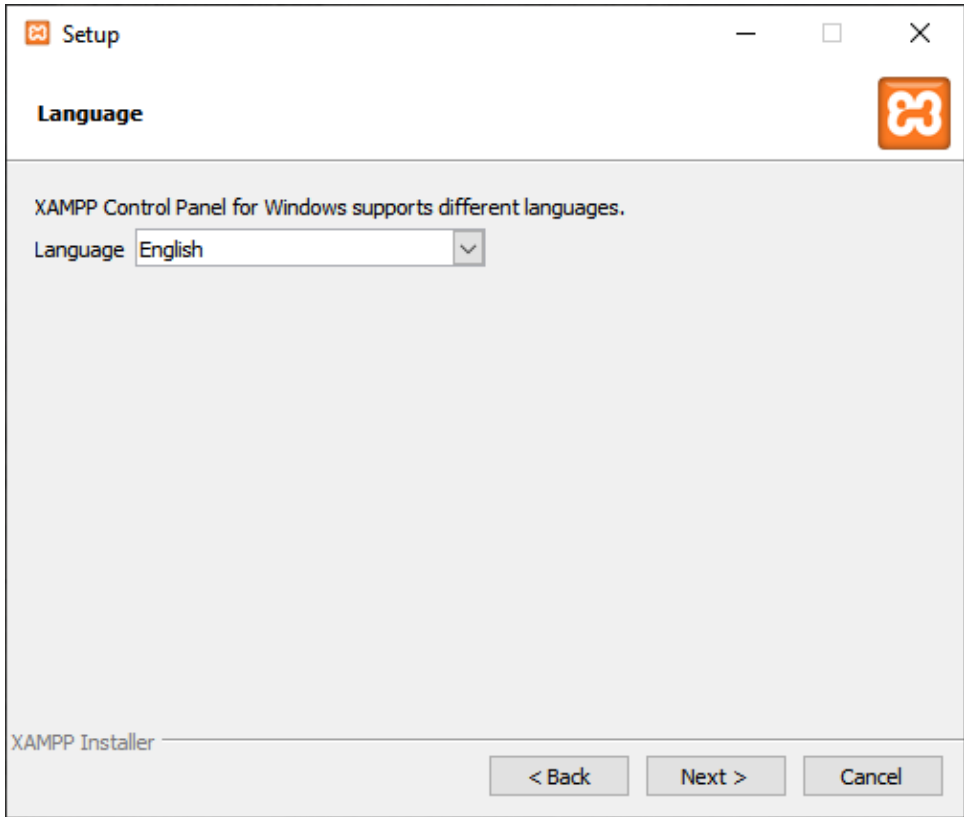
Step 3. Specifying the installation folder

Select a folder to install XAMPP. It's recommended to install XAMPP in the c:\xampp folder. Click the Next button to go to the next step.



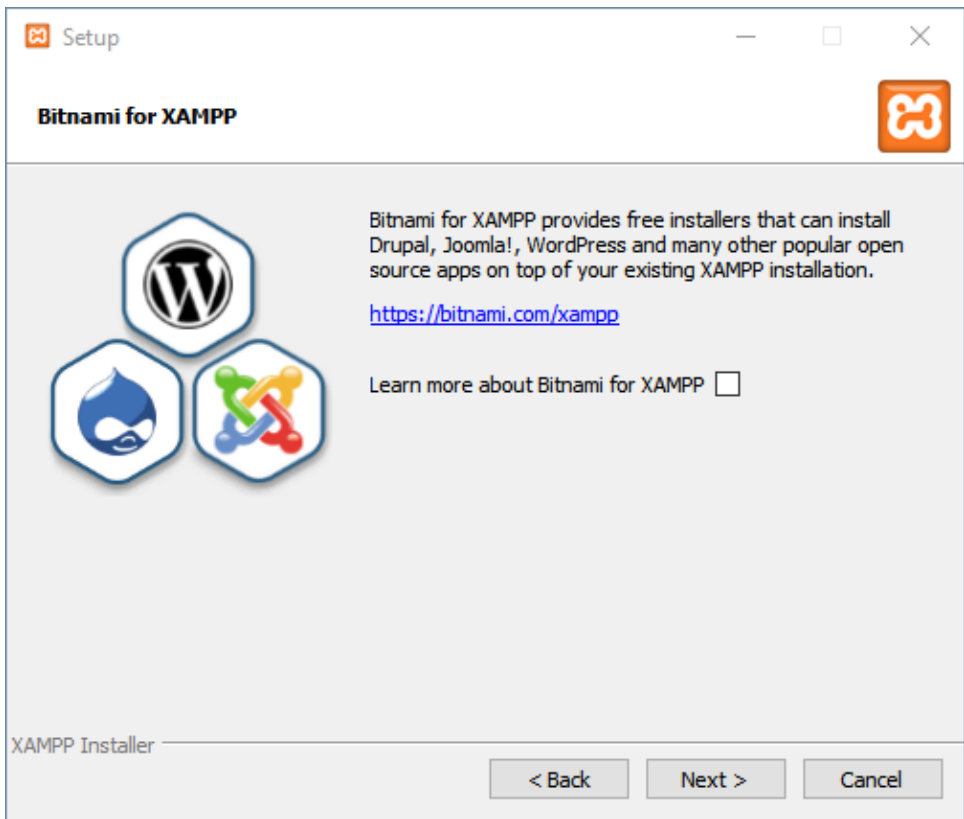
Step 4. Selecting a language

Select a language for the XAMPP Control Panel. By default, it's English. And you can select your preferred language and click the Next button to go to the next step.



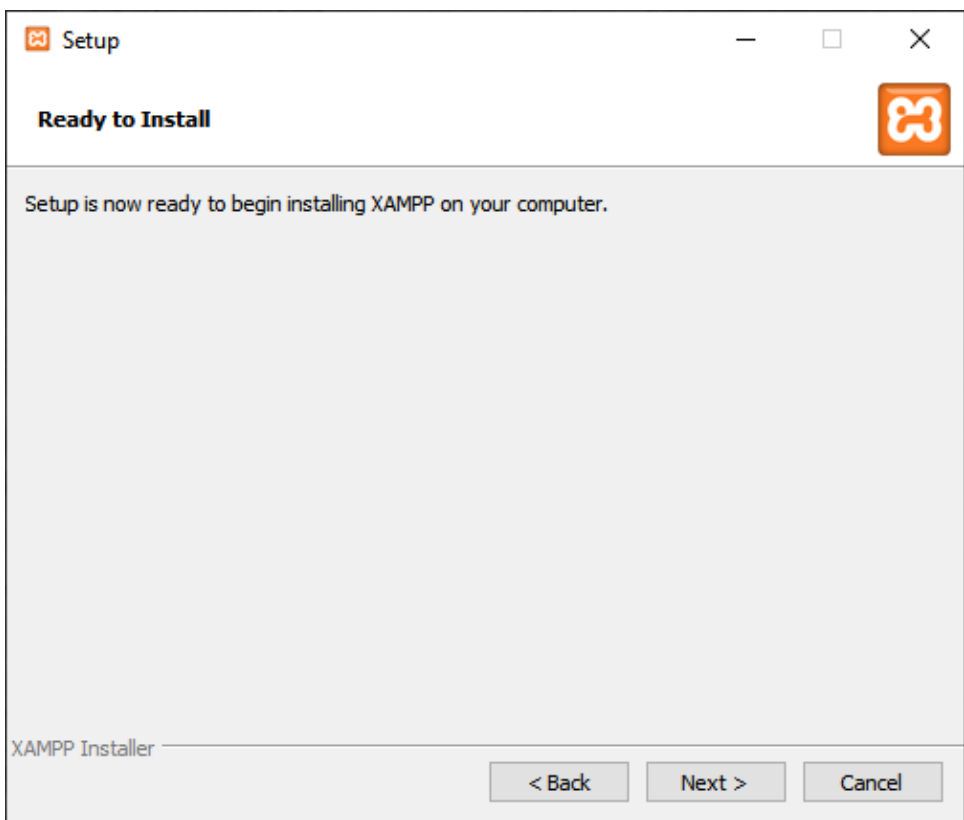
Step 5. Bitnami for XAMPP

Feel free to skip this step because you don't need Bitnami for learning PHP. Just click the Next button to go to the next step.



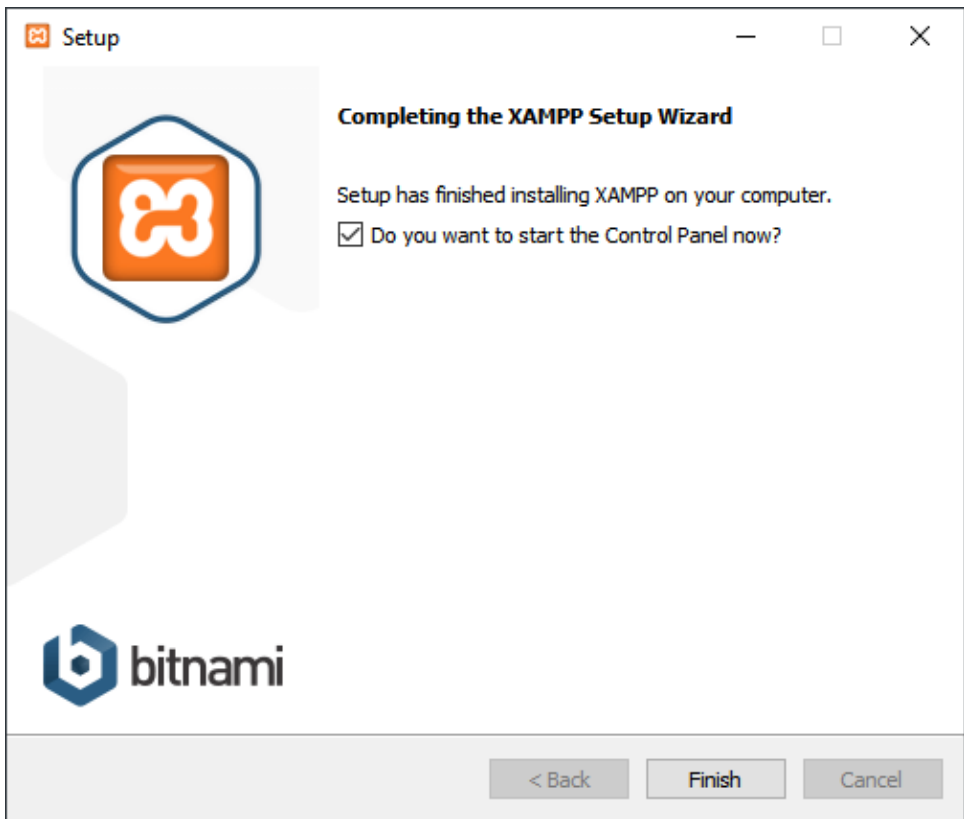
Step 6. Begin installing XAMPP

And you're now ready to install XAMPP. Click the Next button to start the installation. It'll take a few minutes to complete.



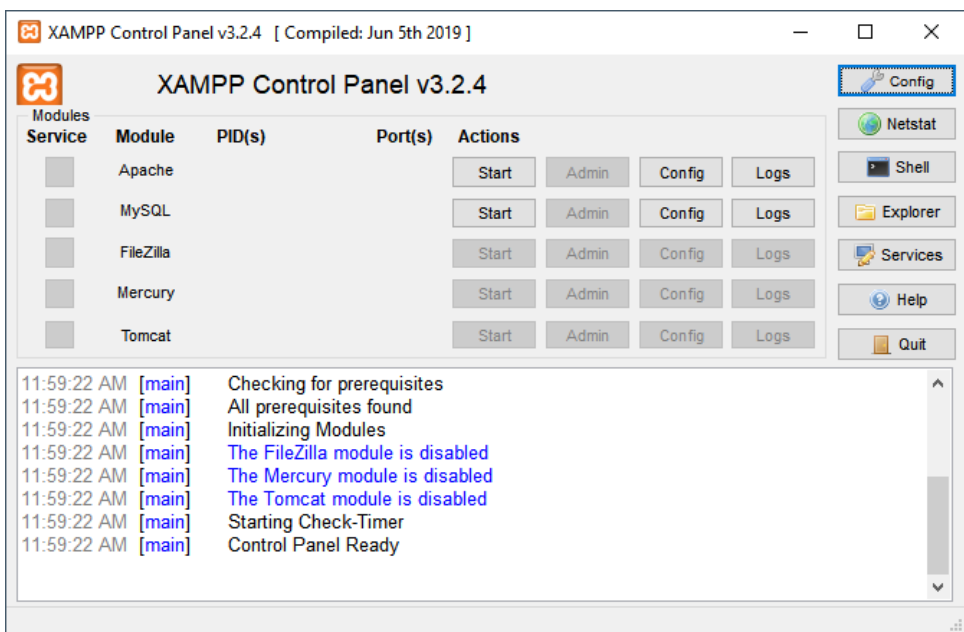
Step 7. Completing the XAMPP setup

Once completed, the XAMPP setup wizard shows the following screen. You can click the Finish button to launch the XAMPP Control Panel:

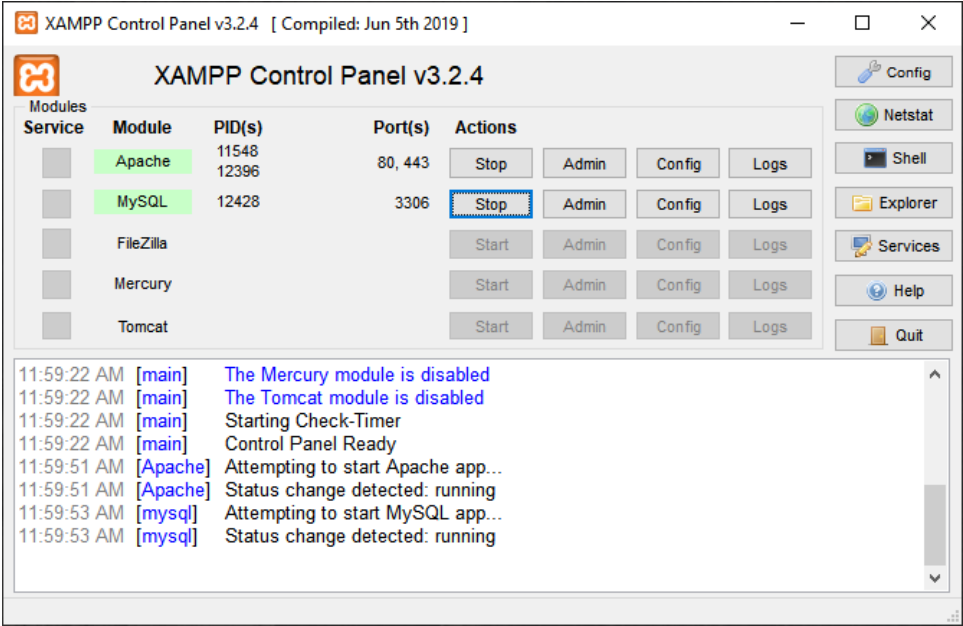


Step 8. Completing the XAMPP setup

The XAMPP Control Panel lists installed services. To start a service, you click the corresponding Start button:



The following shows the Apache web server and MySQL are up and running. The Apache web server listens on the ports 80 and 443 while the MySQL listens on port 3306:



Step 9. Launch the XAMPP

Open the web browser and navigate to the following URL: `http://localhost/`. If the installation is completed successfully, you'll see the welcome screen of the XAMPP.

PHP Script

Each program is called the PHP script. The script is a text file, which can be created using a plain text file editor program such as notepad, edit, vi (Unix / Linux), or else. The text editor used should be a text editor that lets you create a PHP program easier.

PHP Basic Syntax

Like HTML, you need to have the opening tag to start PHP code:

```
<?php
```

If you mix PHP code with HTML, you need to have the enclosing tag:

```
?>
```

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

Generally, each statement is written in a single line. PHP script is a script that is used to generate web pages. How writing can be divided into **embedded** and **non- embedded script**.

Embedded Script and Non-embedded Script

Embedded Script

Following is an example of an HTML document that will be generated using a PHP program/script in an embedded script.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh Embedded</title>
  </head>
  <body>
    <?php
      echo "Hai, saya dari script PHP!";
    ?>
  </body>
</html>
```

Non-Embedded Script

PHP scripts in this way are used as pure programming with PHP, HTML tags generated to create documents are part of the PHP script.

Example

```
<?php
  echo "<html>";
  echo "<head>";
  echo "<title>";
  echo "Non Embedded Script Example";
  echo "</title>";
  echo "</head>";
  echo "<body>";
  echo "<p>Hello, This is the result of Non Embedded Script";
  echo "</body>";
  echo "</html>";
?>
```

Comment

The comment referred to in the PHP script is part of the PHP script that will not be executed, because it is a record of the function of the script or explains the purpose of a portion of the script that is written.

There are 3 ways to write comments in PHP:

1. **/* comment */** - how to write comments from the C programming language
2. **// comment** - how to write comments from the C++ and PHP programming language
3. **# comment** - how to write comments from the Bourne shell script programming language on Unix / Linux

Example

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

```
<?php
// This is a single-line comment

# This is also a single-line comment
?>
</body>
</html>
```

Case Sensitive

PHP scripts apply case sensitive rules namely the differences in writing with uppercase and lowercase letters. Every writing in the script must follow the rules of writing that have been determined, case sensitive is mainly applied to variable names. The following is an example script that shows case sensitive rules.

variables are case-sensitive. e.g., **\$message** and **\$MESSAGE** are different variables.

But,

If you have a function such as **count**, you can use it as **COUNT**. It would work properly.

The following are **case-insensitive** in PHP:

- PHP constructs such as **if, if-else, if-elseif, switch, while, do-while**, etc.
- Keywords such as **true** and **false**.
- User-defined function & class names.

Practice 1!

PHP Hello World! #1

Description

Write a first program to print “Hello World!” using PHP Programming Language.

Steps

Follow the following steps to create a first PHP program:

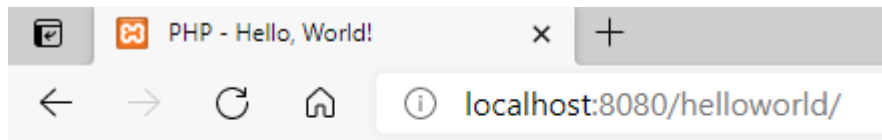
1. open the folder **htdocs** under the xampp folder. Typically, it is located at **C:\xampp\htdocs**.
2. create a new folder called **“helloworld”**.
3. create a new file called **“index.php”** under the helloworld folder and place the following code in the file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>PHP - Hello, World!</title>
</head>
<body>
  <h1><?php echo 'Hello, World!'; ?></h1>
</body>
</html>
```


- launch a web browser and open the URL:

```
http://localhost:8080/helloworld/
```

The result:



Hello, World!



Let's do it!

Make a group with 1-2 members, then do the Exercise below on paper! Submit your work to teacher

Exercise!

- Mention one of the differences of writing PHP script by using embedded script with non embedded script!
- If the program below is executed, what is the output? And how to write the above PHP script using what (*hint: embedded script or embedded script*)?

```
<?php
    $a=false;
    $b=true;
    $k=$a && $b;
    $n=!$k;

    echo "result n is ".$n;
?>
```

- Based on the program below, look for some errors in the writing and fix it, explain!

```
<?php
    $x = 21;
    $w = 3;
    $a = $X - $w;

    echo "result a is ".$a;
?>
```

- What is the HTML tag function: `<body>` `</ body>`?
- What is the PHP script function: `echo`?

CHAPTER 2

PHP FUNDamentals

(Variable, Data Type, Operators)

Define a Variable

A variable stores a value of any type, e.g., a string, a number, an array, or an object. A variable has a name and is associated with a value.

To define a variable, you use the following syntax:

```
$variable_name = value;
```

Rule of PHP Variables:

- The variable name must start with the \$ sign
- The first character after the dollar sign (\$) must be a letter (a-z) or the underscore (_).
- The variable name cannot start with number
- Variable names are case-sensitive (\$age and \$AGE are two different variables)

Example

The following example defines a variable called \$title:

```
<?php
$title = "PHP is awesome!";
?>
```

Example of the Correct Variable

```
<?php
$username
$password
$city
$born_place
?>
```

Example of the Incorrect Variable

```
<?php
$user name
$pass/word
$city 2
$born-place
?>
```

Data Types in PHP

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- Integer
- Float
- String
- Boolean
- Array
- Object
- NULL
- Resource

Integer

An integer data type is a numeric data type used to express integers.

Rules for Integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

Example

The following example defines integer data:

```
<?php
//Integer number
$a = 2;
$b = 3;
$c = $a*$b;
echo $c;
?>
```

Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

Example

The following example defines float data:

```
<?php
//Float number
$a = 2.23;
$b = 3.75;
$c = $a+$b;
echo $c;
?>
```

String

A string is a sequence of characters, like "Hello world!". A string can be any text inside quotes. You can use single or double quotes:

‘Good Morning, World!’ - single quotes

“Good Morning, World!” - double quotation mark

Example

The following example defines string data:

```
<?php
//String Data
$a = "SMA";
$b = "ABBS";

$c = $a." ".$b;

echo $c;
?>
```

Boolean

Boolean data types are data types that are used to declare true or false values, or boolean values. In most programming languages or on the basics of programming, the boolean value is expressed as a basic data type (primitive), namely 1 and 0, 1 to declare true and 0 to declare false, it can also be true or false.

Example

The following example defines boolean data:

```
<?php
//Boolean
$a = true;
$b = false;

?>
```

Operators in PHP

Simple answer can be given using the expression $4 + 5$ is equal to 9. Here 4 and 5 are called **operands** and + is called **operator**. This section covers the most commonly used operators in PHP including logical and comparison operators.

PHP language supports the following type of operators:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators

- Logical operators
- String operators
- Array operators

Arithmetic operators

The arithmetic operators require numeric values. If you apply them to non-numeric values, they’ll convert them to numeric values before carrying out the arithmetic operation.

The following are the list of arithmetic operators:

Operator	Name	Example	Description
+	Addition	\$a + \$b	Return the sum of two operands
-	Subtraction	\$a - \$b	Return the difference between two operands
*	Multiplication	\$a * \$b	Return the product of two operands
/	Division	\$a / \$b	Return the quotient of two operands
%	Modulus	\$a % \$b	Return the remainder of the division of the first operand by the second one

Example

```
<?php
$x = 20;
$y = 10;

// add, subtract, and multiplication operators demo
echo $x + $y . '<br/>'; // 30
echo $x - $y . '<br/>'; // 10
echo $x * $y . '<br/>'; // 200

// modulus demo
$y = 15;
echo $x % $y . '<br/>'; // 5
?>
```

Assignment operators

Assignment operator (=) assigns a value to a variable and returns a value. The operand on the left is always a variable, while the operand on the right can be a literal value, variable, expression, or a function call that returns a value

Besides the basic assignment operator (=), The following are the list of some assignment operators:

Operator	Name	Same as...	Description
a = b	equal	a = b	
a += b	plus-equal	a = a + b	

a -= b	minus-equal	a = a - b	
a *= b	multiplication-equal	a = a * b	
a /= b	divide-equal	a = a / b	

Example

```
<?php

$x = 20;
$y = 10;

// plus-equal operators demo
$x += $y
echo $x;    // 30

?>
```

Comparison operators

Comparison operators allow you to compare two operands. A comparison operator returns a Boolean value, either true or false.

The following are the list of comparison operators in PHP:

Operator	Name	Description
==	Equal to	Return true if both operands are equal; otherwise, it returns false.
===	Identical to	Return true if both operands have the same data type and equal; otherwise, it returns false.
!=	Not equal to	Return true if both operands are equal; otherwise, it returns false.
!==	Not identical to	Return true if both operands are not equal or not have the same data type; otherwise, it returns false.
<	Less than	Return true if the operand on the left is less than the operand on the right; otherwise, it returns false.
>	Greater than	Return true if the operand on the left is greater than the operand on the right; otherwise, it returns false.
<=	Less than or equal	Return true if the operand on the left is less than or equal to the operand on the right; otherwise, it returns false.
>=	Greater than or equal to	Return true if the operand on the left is greater than or equal to the operand on the right; otherwise, it returns false.

Example

```
<?php

$x = 10;
$y = 20;

var_dump($x > $y); // bool(false)
var_dump($y > $x); // bool(true)
var_dump($x == $y); //bool(false)

?>
```

Logical operators

Logical operators allow you to construct logical expressions. A logical operator returns a Boolean value.

The following are the list of logical operators in PHP:

Operator	Name	Description
&&	Logical AND	Return true if both operands are true , otherwise return false. If the first operand is false, it will not evaluate the second operand because it knows for sure that the result is going to be false. This is known as short-circuiting.
	Logical OR	Return true if one of the operands is true , otherwise returns false. If the first operand is true, it will not evaluate the second one.
xor	Logical XOR	Return true if either operand, not both, is true, otherwise, return false.
!	Not	returns true if the operand is false, and returns false if the operand is true.

AND (&&) Operator

The logical AND operator accepts two operands and returns true if both operands are true; otherwise, it returns false.

PHP uses the and / && keyword to represent the logical AND operator:

```
expression1 and expression2
```

The following table illustrates the result of the and operator:

expression1	expression2	expression1 and expression2
true	true	true
true	false	false
false	true	false

false	false	false
-------	-------	-------

Example

```
<?php

$price = 100;
$qty = 5;

$discounted = $qty > 3 && $price > 99;

var_dump($discounted);

?>
```

Output

```
TRUE
```

If you change the \$qty to 1, the \$discounted will be false like this:

```
<?php

$price = 100;
$qty = 1;

$discounted = $qty > 3 && $price > 99;

var_dump($discounted);

?>
```

Output

```
FALSE
```

OR (||) Operator

The logical OR operator accepts two operands and returns true if either operand is true; otherwise, it returns false. In other words, the logical OR operator returns false if both operands are false.

To represent the logical OR operator, PHP uses either the or keyword or the || as follows:

```
expression1 or expression2
```

The following table illustrates the result of the and operator:

expression1	expression2	expression1 or expression2
true	true	true

true	false	true
false	true	true
false	false	false

Example

```
<?php

$price = 100;
$qty = 5;

$discounted = $qty > 3 or $price > 99;

var_dump($discounted);

?>
```

Output

```
TRUE
```

If you change the \$qty to 1, the \$discounted still be true as well like this:

```
<?php

$price = 100;
$qty = 1;

$discounted = $qty > 3 && $price > 99;

var_dump($discounted);

?>
```

Output

```
TRUE
```

Concatenating Operator

Concatenating operator (.) allows you to **combine two strings into one**. It appends the second string to the first one and returns the combined string. For example:

```
<?php

$str = 'Selamat Pagi' . ' is ' . ' Good Morning!';
echo $str;

?>
```

Output

Selamat Pagi is Good Morning!

Practice 2!

General Steps

1. open the folder `htdocs` under the `xampp` folder. Typically, it is located at `C:\xampp\htdocs`.
2. Create a new folder called “**chapter2**” and you can use the folder to store your practice!

PHP Variable #1

Description

Write a program to print 2 php variables using a single echo statement.

Steps

Follow the following steps to create a program:

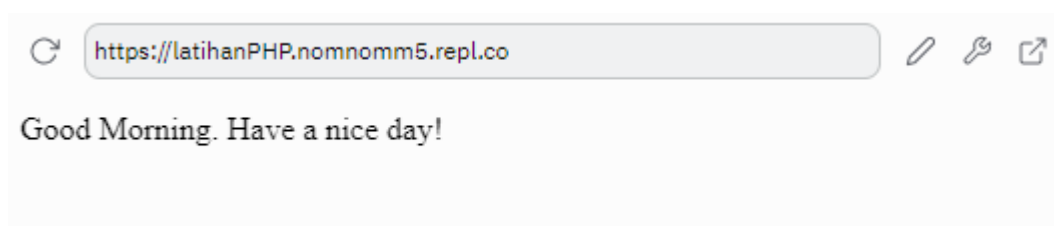
1. Create a new file called “`practice_1.php`” under the `chapter_2` folder
2. Write the following code in the file:

```
<?php
$message_1 = "Good Morning.";
$message_2 = "Have a nice day!";
echo $message_1." ". $message_2;
?>
```

3. Launch a web browser and open the URL:

`http://localhost/chapter_2/practice_1.php`

The result:



PHP Variable #2

Description

Write a program to sum variables using a single echo statement.

Steps

Follow the following steps to create a program:

1. Create a new file called “`practice_2.php`” under the `chapter_2` folder
2. Write the following code in the file:

```
<?php
$a = 10;
$b = 4;
```

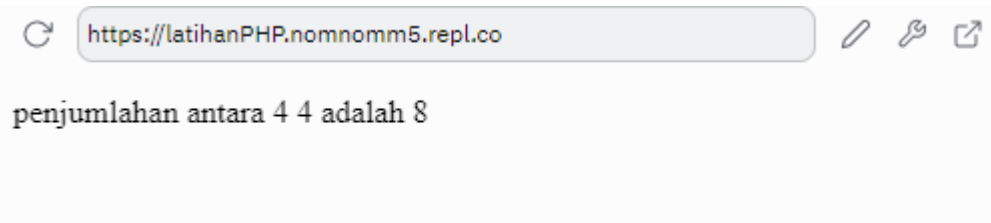
```
$a= $b;
$c = $a + $b;

echo "penjumlahan antara ".$a. " " . $b. " adalah ".$c;
?>
```

3. Launch a web browser and open the URL:

```
http://localhost/chapter_2/practice_2.php
```

The result:



penjumlahan antara 4 4 adalah 8

PHP Variable #3

Description

Write a program to sum variables using a single echo statement.

Steps

Follow the following steps to create a program:

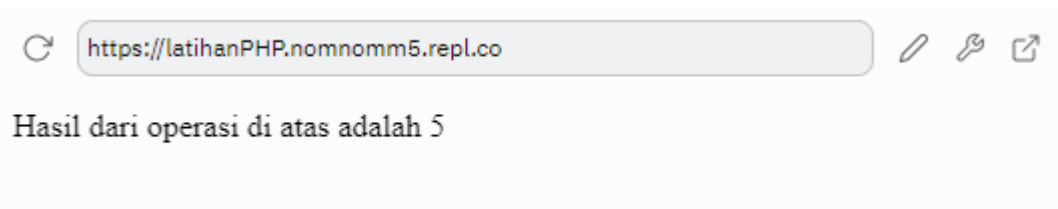
1. Create a new file called “practice_3.php” under the chapter_2 folder
2. Write the following code in the file:

```
<?php
$x=10;
$y=4;
$z=3;
echo "Hasil dari operasi di atas adalah ".$x % ($y) + $z);
?>
```

3. Launch a web browser and open the URL:

```
http://localhost/chapter_2/practice_3.php
```

The result:



Hasil dari operasi di atas adalah 5

PHP Operators#4

Description

Write a program to print “Good morning, world!” using logical operator

Steps

Follow the following steps to create a program:

1. Create a new file called “practice_4.php” under the chapter_2 folder

2. Write the following code in the file:

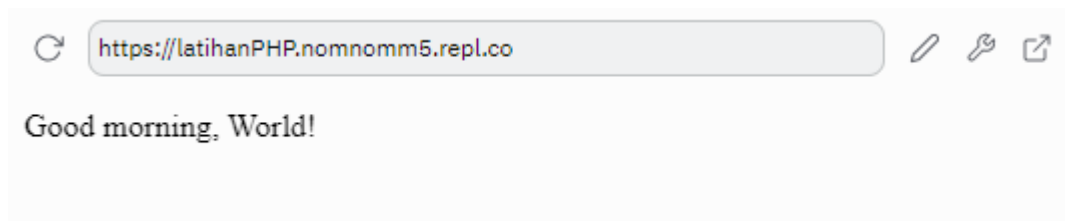
```
<?php
$x = 100;
$y = 50;

if ($x == 100 && $y == 50) {
    echo "Good morning, world!";
}else{
    echo "Good morning, Afternoon!";
}
?>
```

3. Launch a web browser and open the URL:

http://localhost/chapter_2/practice_4.php

The result:



Exercise!

- Determine right or wrong for some of the variables below!
 - \$123user
 - \$first name
 - \$data_siswa
 - \$data2
- Based on the program below, what types of operators are included in the program above?

```
<?php
$a=12;
$b=7;
$c=$a-$b;
?>
```

3. What is the result of \$p?

```
<?php
$k=(4-1)*2;
$n=12-$k;
$p=93%n;
?>
```

4. If \$ a = 1, what is the final output of the program above?

```
<?php
$a= ;
if($a<=0){
    echo "False";
}else{
```

```
        echo "True";  
    }  
    echo "Result a is : ".$a;  
?>
```

5. If the program is executed, then the output of \$ k is

```
<?php  
$x=12;  
$y=5;  
$z=4+$x<20;  
$c=$y+1<$z;  
$k=$z && $c;  
?>
```

CHAPTER 3

Control Structure

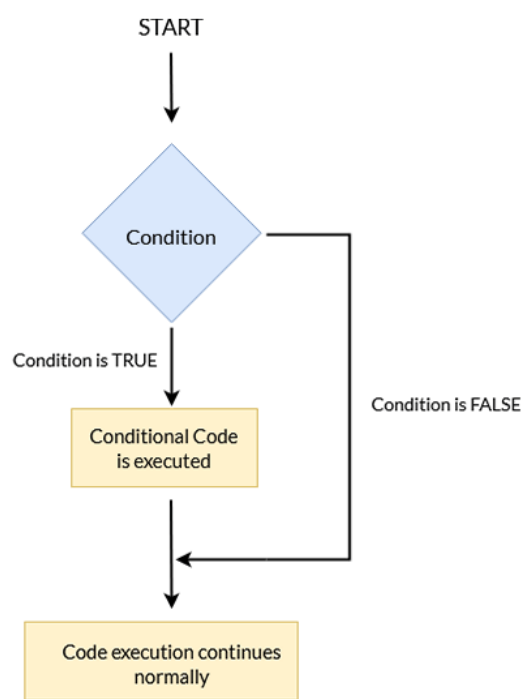
(Condition & Iteration)

What is Control Structure?

A control structure, in simple terms, allows you to control the flow of code execution in your application. A program is typically executed sequentially, line by line, and a control structure allows you to change that flow, usually based on certain conditions.

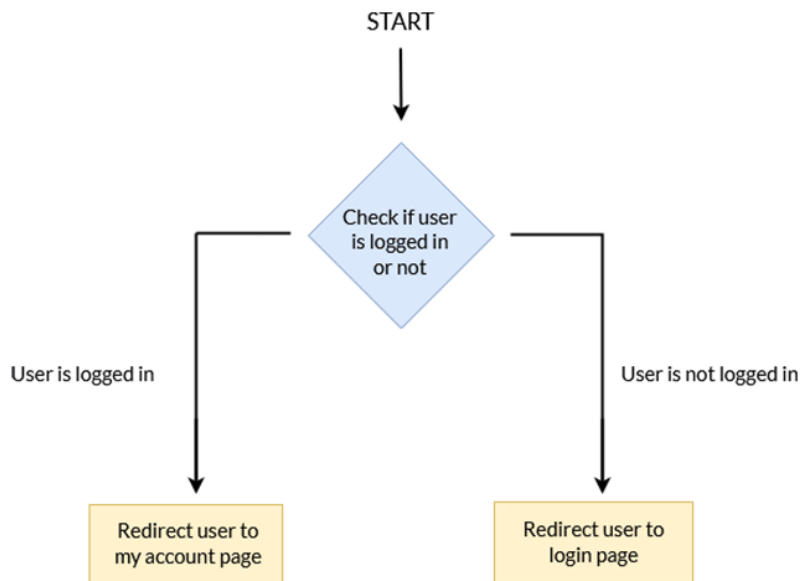
Control structures are PHP language features that enable your script to respond differently to different inputs or situations. This allows your script to respond differently based on user input, file contents, or other data.

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



As you can see in the above diagram, 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.

In PHP, there are two primary types of Control Structures:

- Conditional Statements
 - If Statements
 - Else Statements
 - Else If Statements
 - Switches
- Control Loops
 - While Loops
 - Do.. While Loops
 - For Loops
 - Foreach Loops

Conditional Statements

Conditional Statements allow you to branch the path of execution in a script based on whether a single or multiple conditions evaluate to true or false. Simply put, they allow you to test things and then take various actions based on the results.

IF Statements

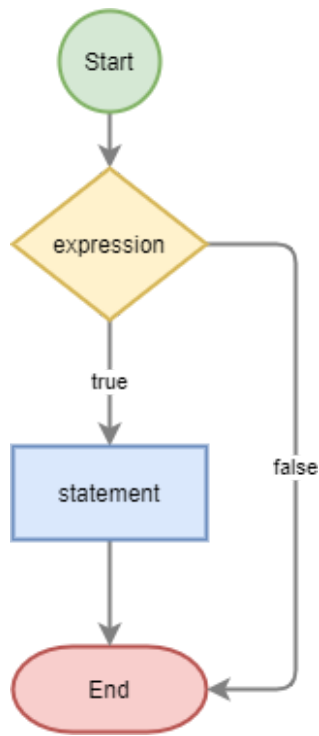
The IF statement allows you to execute a statement if an expression evaluates to true.

Syntax

```
if (expression) {  
    statement - code to be executed if expression is true;  
}
```

In this syntax, PHP evaluates the expression first. If the expression evaluates to true, PHP executes the statement. In case the expression evaluates to false, PHP ignores the statement.

The following flowchart illustrates how the if statement works:



Example

The following example uses the if statement to display a message if the \$numb variable greater than 10:

```
<?php
$numb = 20;

if ($numb > 10) {
    echo "Have a good day!";
}
?>
```

Output

```
Have a good day!
```

ELSE Statements

The IF ELSE statement allows you to execute a statement if an expression evaluates to true and another expression is false.

Syntax

```
if (expression) {
    statement - code to be executed if expression is true;
} else {
    statement - code to be executed if expression is false;
```

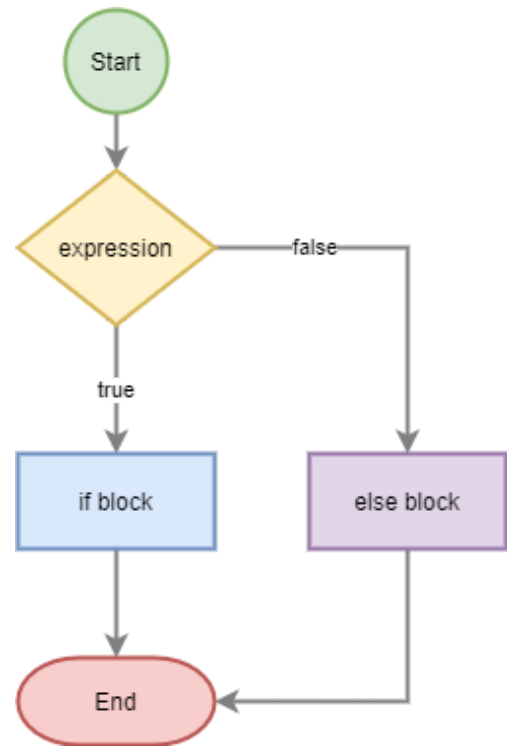


```
}

```

In this syntax, if the expression is true, PHP executes the code block that follows the if clause. If the expression is false, PHP executes the code block that follows the else keyword.

The following flowchart illustrates how the PHP IF ELSE statement works:



Example

The following example uses the ELSE statement to display a message show a message based on the value of the \$numb variable:

```
<?php
$numb = 5;

if ($numb > 10) {
    echo "Have a good day!";
}
else {
    echo "Have a good night!";
}
?>

```

Output

```
Have a good night!

```

ELSE IF Statements

The IF statement can have one or more optional ELSE IF clauses. The ELSE EF is a combination of if and else. The ELSE IF statement executes different codes for more than two expressions.

Syntax

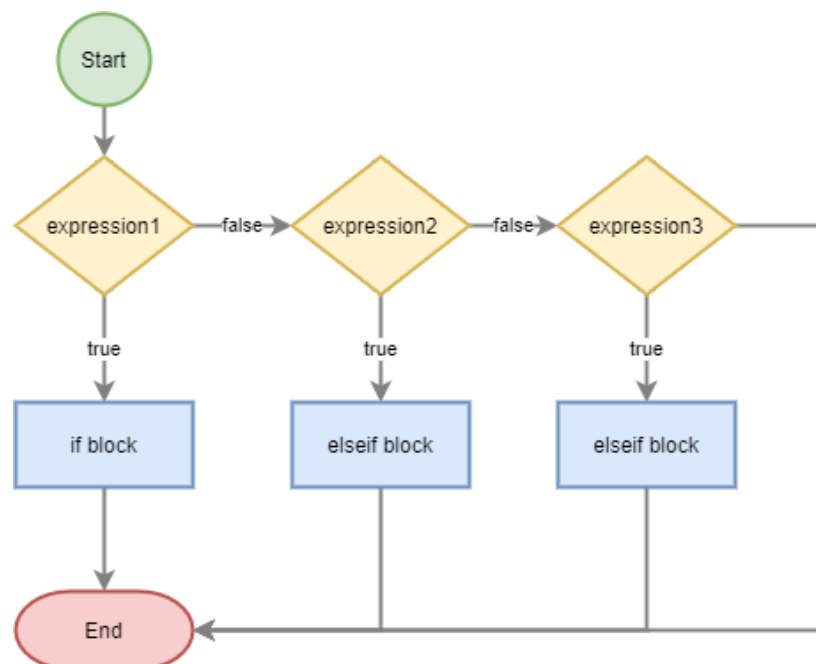
```
if (expression1) {  
    statement1 - code to be executed if this expression1 is true;  
} elseif (expression2) {  
    statement2 - code to be executed if first expression1 is false  
    and this expression2 is true;  
} else {  
    statement3 - code to be executed if all expressions are false;  
}
```

PHP evaluates the expression1 and executes the code block in the if clause if the expression1 is true.

If the expression1 is false, the PHP evaluates the expression2 in the next ELSE IF clause. If the result is true, then PHP executes the statement in that elseif block. Otherwise, PHP evaluates the expression3.

If the expression3 is true, PHP executes the block that follows the ELSE IF clause. Otherwise, PHP ignores it.

The following flowchart illustrates how the ELSE IF statement works:



Example

The following example uses the ELSE statement to display a message show a message based on the value of the \$numb variable:

```
<?php  
$numb1 = 5;  
$numb2 = 10;  
  
if ($numb1 > $numb2) {  
    echo "numb1 is greater than numb2 ";  
} elseif ($numb1 < $numb2) {  
    echo "numb1 is less than numb2";  
}
```

```
}else{  
    echo "numb1 is equal to numb2";  
}  
?>
```

Output

```
numb1 is less than numb2
```

SWITCHES

When the value of a single variable determines the number of different choices, you can use the ELSE IF statement. Suppose that you're building a website whose users have many roles like admin, editor, author, and subscriber.

When the value of a single variable specifies the number of different choices, it's much cleaner to use the SWITCH statement.

Syntax

```
switch (expression) {  
    case value1:  
        // code block 1  
        break;  
    case value2:  
        // code block 2  
        break;  
    case value3:  
        // code block 3  
        break;  
    default:  
        // default code block  
}
```

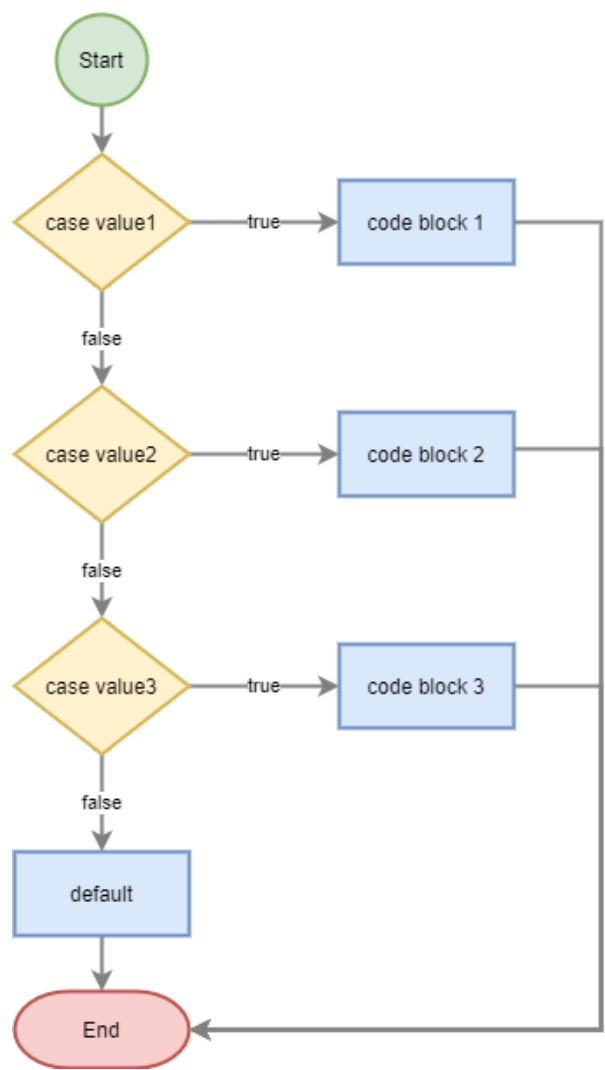
The switch statement compares an expression with the value in each case.

If the expression equals a value in a case, e.g., value1, PHP executes the code block in the matching case until it encounters the first break statement.

If there's no match and the default is available, PHP executes all statements following the default keyword.

In case the default is not specified, and there's no match, the control is passed to the statement that follows the switch statement.

The following flowchart illustrates how the SWITCH statement works:



Example

The following example uses the ELSE statement to display a message show a message based on the value of the \$numb variable:

```
$status  = "admin";

switch ($status) {
    case "student":
        echo "Welcome student! Check out your Final Result.";
        break;
    case "admin":
        echo "Welcome Admin! How are you today?";
        break;
    default:
        echo "You are not authorized to access this page!";
}
```

Output

```
Welcome Admin! How are you today?
```

Control Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

FOR Statements

The for statement allows you to execute a code block repeatedly and can be determined how many times it must be done.

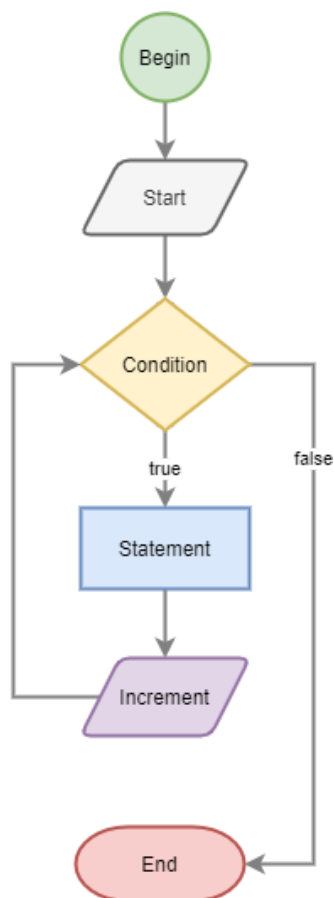
Syntax

```
for (start; condition; increment) {  
    statement;  
}
```

How it works:

- The start is evaluated once when the loop starts.
- The condition is evaluated once in each iteration. If the condition is true, the statement in the body is executed. Otherwise, the loop ends.
- The increment expression is evaluated once after each iteration.

The following flowchart illustrates how the PHP FOR statement works:



Example

The following example uses the ELSE statement to display a message show a message based on the value of the \$numb variable:

```
<?php
for ($x = 0; $x <= 5; $x++) {
    echo "The number is: $x <br>";
}
?>
```

Output

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

WHILE Statements

The while statement executes a code block as long as an expression is true. The syntax of the while statement is as follows

Syntax

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

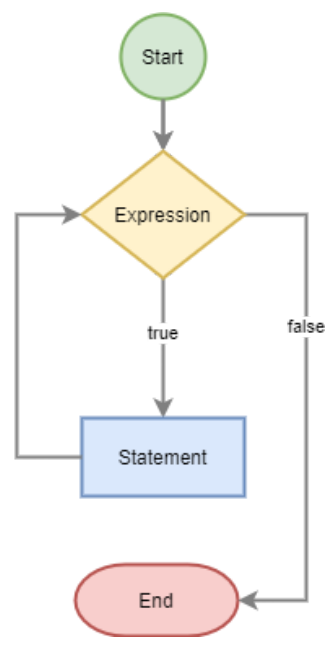
How it works:

- First, PHP evaluates the expression. If the result is true, PHP executes the statement.
- Then, PHP re-evaluates the expression again. If it's still true, PHP executes the statement again. However, if the expression is false, the loop ends.

If the expression evaluates to false before the first iteration starts, the loop ends immediately.

Since PHP evaluates the expression before each iteration, the while loop is also known as a **pretest loop**.

The following flowchart illustrates how the PHP WHILE statement works:



Example

The following example uses a while loop to add whole numbers from 1 to 10:

```
<?php
$total = 0;
$number = 1;

while ($number <= 10) {
    $total += $number;
    $number++;
}

echo $total;
?>
```

Output

```
55
```

DO-WHILE Statements

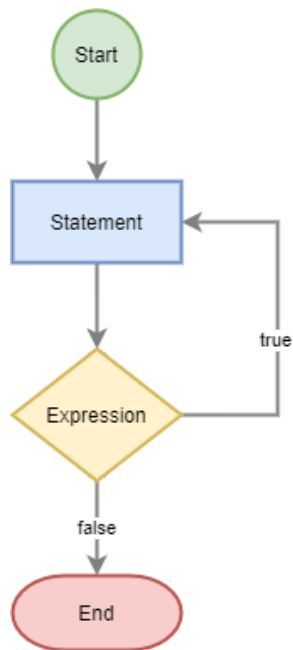
The PHP DO-WHILE statement allows you to execute a code block repeatedly based on a Boolean expression. This loop will always execute the block of code once, it will then check the expression, and repeat the loop while the specified expression is true.

Syntax

```
do {
    statement;
} while (expression);
```

Unlike the while statement, PHP evaluates the expression at the end of each iteration. It means that the loop always executes at least once, even if the expression is false before the loop enters.

The following flowchart illustrates how the PHP DO-WHILE statement works:



Difference between DO-WHILE and WHILE

The differences between the do...while and while statements are:

- PHP executes the statement in do...while at least once, whereas it won't execute the statement in the while statement if the expression is false.
- PHP evaluates the expression in the do...while statement at the end of each iteration. Conversely, PHP evaluates the expression in the while statement at the beginning of each iteration.

Example

The following example uses a while loop to add whole numbers from 1 to 10:

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

Output

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```


Practice 3!

General Steps

1. open the folder `htdocs` under the `xampp` folder. Typically, it is located at `C:\xampp\htdocs.`
2. Create a new folder called “**chapter3**” and you can use the folder to store your practice!

Condition Statement #1

Description

Write a PHP script to determine odd and even numbers from an input variable.

Steps

Follow the following steps to create a program:

1. Create a new file called “**condition.php**” under the **chapter3 folder**
2. Write the following code in the file:

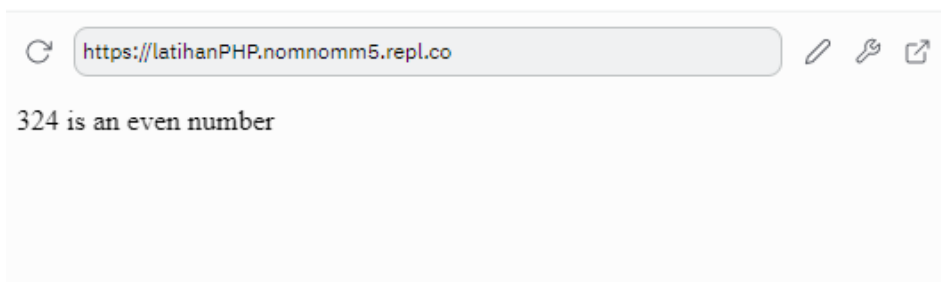
```
<?php
$number = 324; // enter your presence number

if ($number%2 == 0) {
    echo $number . " is an even number";
} else if ($number%2 == 1) {
    echo $number . " is a odd number ";
} else {
    echo " please enter a numeric value";
}
?>
```

3. Launch a web browser and open the URL:

`localhost/chapter3/condition.php`

The result:



Switch Break Statement #2

Description

Write a PHP script to check the arithmetic operations.

Steps

Follow the following steps to create a program:

1. Create a new file called “**switch.php**” under the **chapter3 folder**
2. Write the following code in the file:

```

<?php
$check = 'subtract';

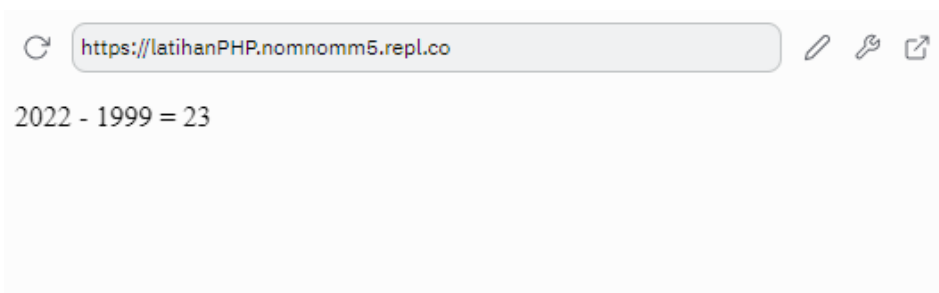
switch ($check) {
    case 'add':
        $a = 10 + 40;
        echo "10 + 40 = $a";
        break;
    case 'subtract':
        $b = 2022 - 1999;
        echo "2022 - 1999 = $b";
        break;
    case 'multiple':
        $c = 76 * 4;
        echo "76 x 4 = $c";
        break;
    case 'divide':
        $d = 54/3;
        echo "54 / 3 = $d";
        break;
    default:
        echo "Incorrect variable";
        Break;
}
?>

```

3. Launch a web browser and open the URL:

localhost/chapter3/switch.php

The result:



While Statement #3

Description

Write a Program to display count, from 5 to 15 using PHP loop as given below.

Steps

Follow the following steps to create a first PHP program:

1. Create a new file called **“while.php”** under the **chapter3 folder**
2. Write the following code in the file:

```

<?php
$count = 5;

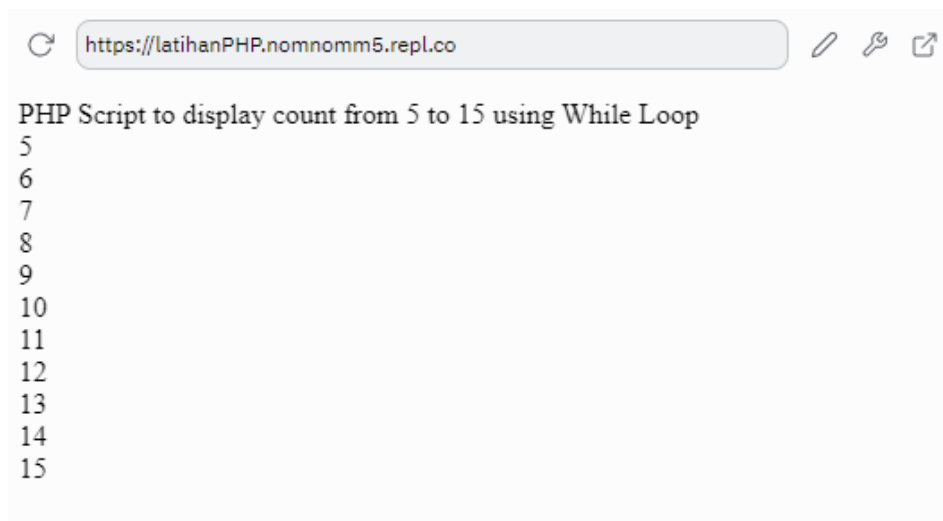
```

```
echo "PHP Script to display count from 5 to 15 using While Loop  
<br>";  
while($count<=15){  
    echo $count;  
    echo "<br> ";  
  
    $count++;  
}  
?>
```

3. Launch a web browser and open the URL:

localhost/chapter3/while.php

The result:



FOR Statement #4

Description

Write a PHP script to add all the integers between 1 and 17 and display the total.

Steps

Follow the following steps to create a first PHP program:

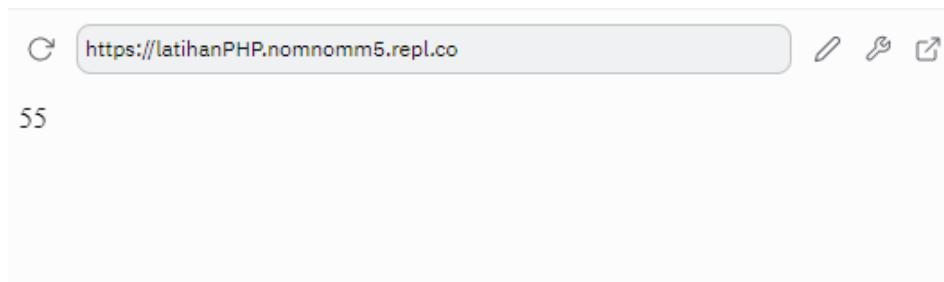
4. Create a new file called **“for.php”** under the **chapter3 folder**
5. Write the following code in the file:

```
<?php  
$total = 0;  
  
for ($i = 1; $i <= 10; $i++) {  
    $total += $i;  
}  
echo $total;  
?>
```

6. Launch a web browser and open the URL:

localhost/chapter3/for.php

The result:



Nested Loop and Condition Statement #5

Description

Write a PHP script to add all the integers between 0 and 30 and display the total.

Steps

Follow the following steps to create a first PHP program:

7. Create a new file called “**nested.php**” under the **chapter3** folder
8. Write the following code in the file:

```
<?php
$count = 1;
$sum = 0;
while($count<=10){
    if($count < 10){
        echo "$count + ";
    }else{
        echo "$count = ";
    }
    $sum += $count;

    $count++;
}

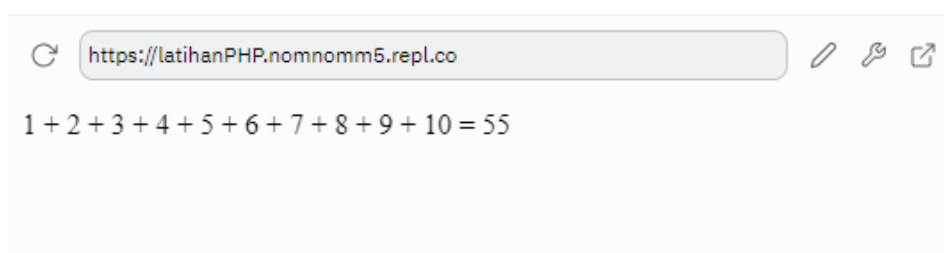
echo $sum;

?>
```

9. Launch a web browser and open the URL:

localhost/chapter3/nested.php

The result:



Exercise 3!

1. What is the result of the script below?

```
<?php
$data1 = (7-4)<5;
$data2 = (8%2)<=2;
if($data==1){
    for($i = 0; $i<10; $i++){
        echo $i;
    }
}else{
    $count = 0;
    while($count<=5){
        echo $count;
        $count++;
    }
}
?>
```

2. Write the PHP script using the Nested FOR and ELSE IF statement to display Roman numerals 1 to 5 of a number!

Output:
I II III IV V

3. What is the result of the script below?

```
<?php
$abc = 5;
do{
    echo "$abc";
    echo " ";
    $abc = $abc+5;
}while($abc<=50)
?>
```

4. Mention and explain the Loops Type in PHP?