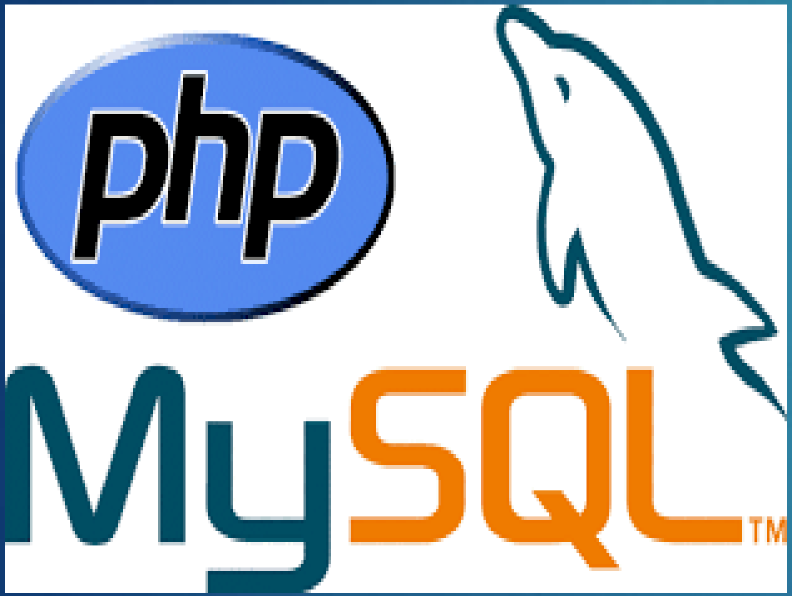
Introduction to php MySQL™

PROF. DR. K. ADISESHA

PHP-Programming



PHP-Programming Part-1

Introduction

Introduction to PHP:

PHP: Hypertext Preprocessor (originally named Personal Home Page Tools). Invented by Rasmus Lerdorf in 1994 and is now under the Apache Software Foundation.

- PHP is an recursive acronym for "*PHP: Hypertext Preprocessor*"
- Popular server-side technology for Apache web servers. Competing technologies include Oracle's JavaServer Pages, Microsoft's ASP.NET, and Adobe's ColdFusion.
- Available on a variety of web servers (Apache, IIS, NGINX, etc.) and operating systems (Windows, Linux, UNIX, Mac OS X, etc.).
- Supports many types of databases: MySQL, Oracle, ODBC (for MS Access and SQL Server), SQLite, etc.

Introduction

History of PHP:

PHP was developed by Rasmus Lerdorf for monitoring his online resume in 1995. It slowly became popular. This first version PHP/F1 has some basic functions.

- PHP/F1 2.0 was released and was quickly replaced by PHP 3.0 in 1997
- PHP 3.0 was developed by and Andi Gutmans and Zeev suraski. It includes support for wider range of data bases (Oracle and My SQL).
- PHP 4.0 was released in 2003. It supports OOP features and built-in session management.
- PHP 5.0 It also includes better exception handling, a more consistent XML tool kit, improved My SQL support and a better memory manager.
- PHP 7 is much faster than the previous popular stable release (PHP 5.6). It has improved Error Handling.

Introduction

Features of PHP programming:

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

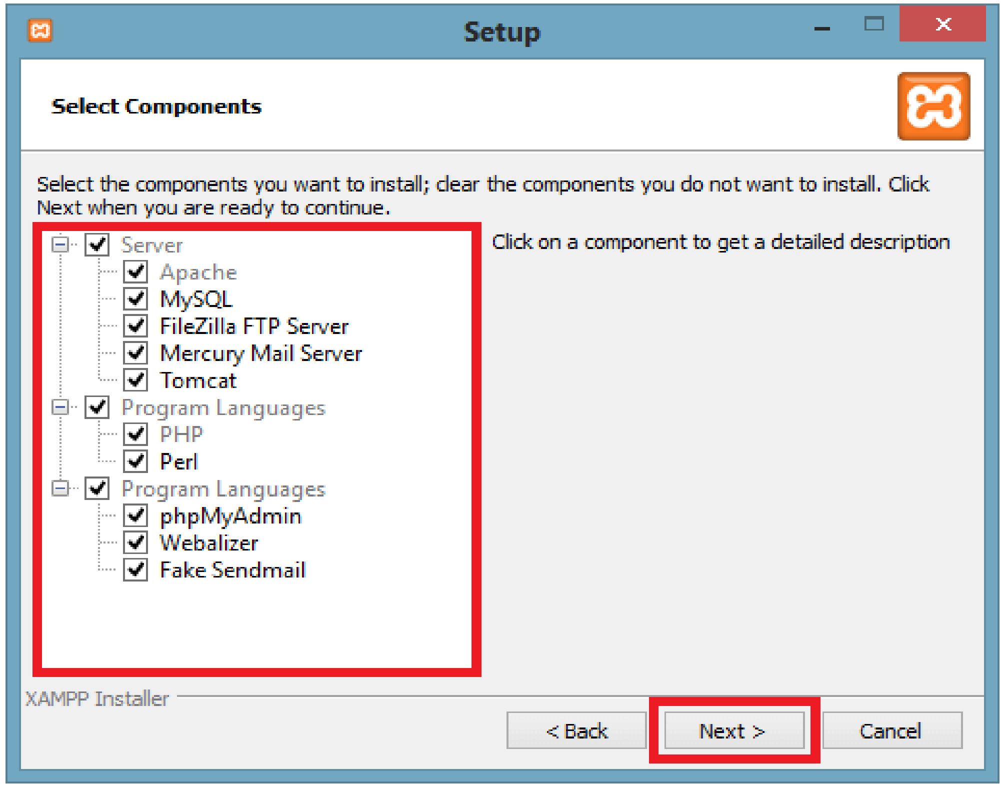
- *PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.).*
- **There are the following features of PHP programming:**
    - *PHP can generate dynamic page content*
    - *PHP can create, open, read, write, delete, and close files on the server*
    - *PHP can collect form data*
    - *PHP can send and receive cookies*
    - *PHP can add, delete, modify data in your database*
    - *PHP can be used to control user-access*
    - *PHP can encrypt data*

Introduction

Installation of PHP:

To install PHP, I will suggest you to install XAMP (Apache, MySQL, PHP) software stack. It is available for all operating systems.

- Download and *Install XAMPP Server*, double click on the downloaded file and allow XAMPP to make changes in your system.
- Here, select the components, which you want to install and click Next.
- Choose a folder where you want to install the XAMPP in your system and click Next.
- XAMPP is ready to install, so click on the Next button and install the XAMPP.


Setup

Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Server
  - [✓] Apache
  - [✓] MySQL
  - [✓] FileZilla FTP Server
  - [✓] Mercury Mail Server
  - [✓] Tomcat
- [✓] Program Languages
  - [✓] PHP
  - [✓] Perl
- [✓] Program Languages
  - [✓] phpMyAdmin
  - [✓] Webalizer
  - [✓] Fake Sendmail

Click on a component to get a detailed description

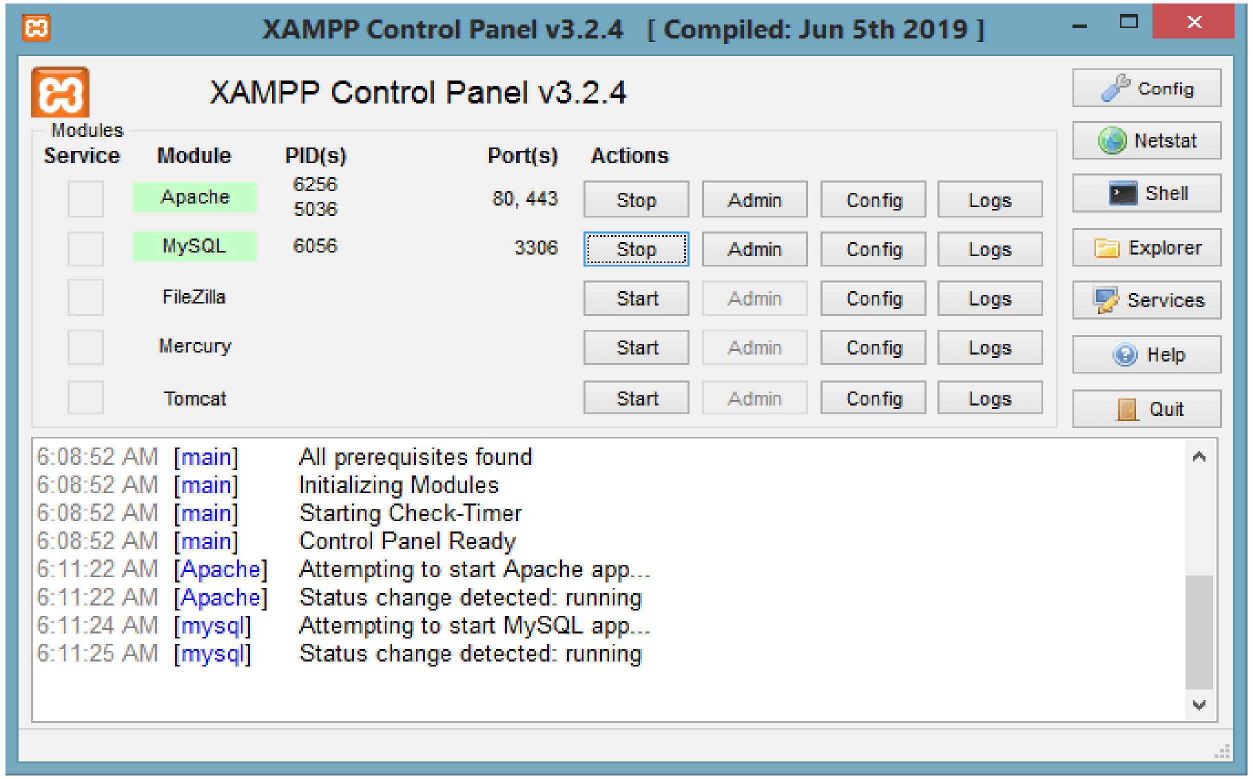
XAMPP Installer

Introduction

PHP programs on XAMPP:

XAMPP is ready to use. Start the Apache server and MySQL and run the PHP program on the localhost.

- ➢ PHP file contains HTML tags and some PHP scripting code and save this file with .php extension.
- ➢ Now, open the web browser and type localhost http://localhost/file.php on your browser window.
- ➢ The output for the above file.php program will be shown in web browser.


XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]
XAMPP Control Panel v3.2.4
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	6256 5036	80, 443	Stop	Admin	Config	Logs
	MySQL	6056	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
6:08:52 AM  [main]     All prerequisites found
6:08:52 AM  [main]     Initializing Modules
6:08:52 AM  [main]     Starting Check-Timer
6:08:52 AM  [main]     Control Panel Ready
6:11:22 AM  [Apache]   Attempting to start Apache app...
6:11:22 AM  [Apache]   Status change detected: running
6:11:24 AM  [mysql]    Attempting to start MySQL app...
6:11:25 AM  [mysql]    Status change detected: running
```

Introduction

Running a PHP script:

PHP code should be placed between <? code ?> or <?php …code ?> tags. The second method is preferred so your scripts are XML compatible. There is no limitation as to where PHP code can be inserted.

1. *Start Windows*
2. *Click start –program files-Accessories-notepad*
3. *Start typing necessary – commands/coding*
4. *Click file – Save as*
5. *Type filename with 'PHP extension'*
6. *Click save button*
7. *Start PHP interpreter.*
8. *Run in browser: type localhost http://localhost/Welcome.php*

Example:

Welcome.php

```
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

PHP Programming

PHP Statements and Comments

A PHP script consists of one or more statements, with each statement ending with a semicolon. Blank lines within the script are ignored by the parser.

- The PHP code is written between the tags <? PHP ……. ?> is read and executed. The semicolon can be omitted on the last line of a PHP block, because the closing ?>
- In PHP, keyword (e.g., echo, if, else, while), functions, user-defined functions, classes are not case-sensitive. However, all variable names are case-sensitive.
- Comments: There are three comment styles listed here:
  - *// this is a single-line comment*
  - *# so is this*
  - */* and this is a multiline comment */*

PHP Programming

Variables in PHP:

Variables are the building blocks of any programming language. A variable can be a programming construct used to store both numeric and nonnumeric data.

- ➢ Every variable has a name, which is preceded by a dollar ($) symbol.
- ➢ It must begin with a letter or underscore character, optionally followed by more letters, numbers, and underscores.
- ➢ Variables need not be declared and are case-sensitive
- ➢ PHP supports a number of different variable types—*Booleans, integers, floating point numbers, strings, arrays, objects, resources, and NULL.*

❖ **For example:** *$bca, $BCA_course, and $COURSE are all valid PHP variable names, $123 and $48hrs are invalid variable names.*

PHP Programming

Variables in PHP:

A variable can have a short name (like $x and $y) or a more descriptive name ($age, $carname, $total_volume).

- The PHP *echo* statement is often used to output data to the screen: *echo "Hello";*
- The *print* statement can be used with or without parentheses: *print or print().*
- **Rules for PHP variables:**
  - *A variable starts with the $ sign, followed by the name of the variable*
  - *A variable name must start with a letter or the underscore character*
  - *A variable name cannot start with a number*
  - *A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)*
  - *Variable names are case-sensitive ($age and $AGE are two different variables)*

PHP Programming

Variables Scope in PHP:

The scope of a variable is the part of the script where the variable can be referenced/used.

➢ PHP has three different variable scopes:

❖ **Global:** *A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function*

❖ **Local:** *A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function*

❖ **Static:** *when a function is completed/executed, all of its variables are deleted. However, to retain a local variable from deletion we need use the static keyword when you first declare the variable.*

```
<?php
$x = 5; // global scope
$y = 10; // global scope
function myTest() {
static $x=20; // static scope
        $y=30; // local scope
  $y = $x + $y;
  echo $x; // out put static variable $x =20
  echo $y; // out put local variable $y =50
}
myTest();  // run function
echo $y; // output the global variable $y =10
?>
```

PHP Programming

PHP Data Types

A PHP script consists Variables can store data of different types, and different data types can do different things.

- ➢ You can get the data type of any object by using the ***var_dump()*** function.
- ➢ PHP supports the following data types:
  - ❖ *String : A string can be any text inside single or double quotes.*
  - ❖ *Integer: An integer data type is a non-decimal number.*
  - ❖ *Float: A float is a number with a decimal point or a number in exponential form.*
  - ❖ *Boolean: A Boolean represents two possible states: TRUE or FALSE.*
  - ❖ *Array: An array stores multiple values in one single variable.*
  - ❖ *Object: An object is an instance of a class*
  - ❖ *NULL: Null is a special data type which can have only one value: NULL.*
  - ❖ *Resource: It is the storing of a reference to functions and resources external to PHP.*

PHP Programming

PHP Data Types

String : A string can be any text inside single or double quotes.

- ➢ A string is a sequence of characters, like "Hello world!".
- ➢ **Various String Functions:**
  - ❖ *strlen() function returns the length of a string.*
  - ❖ *strrev() function reverses a string*
  - ❖ *strpos() function searches for a specific text within a string.*
  - ❖ *str_word_count() function counts the number of words in a string*
  - ❖ *str_replace() function replaces some characters with some other characters in a string.*

```
<html>
<body>
<?php
echo strlen("Hello world!");
echo strrev("Hello world!");
echo strpos("Hello world!", "world");
echo str_word_count("Hello world!");
echo str_replace("world", “Adi", "Hello world!");
?>
</body>
</html>
```

PHP Programming

PHP Data Types

***PHP Integer:* An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.**

- **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 three formats:*
    - *decimal (10-based),*
    - *hexadecimal (16-based - prefixed with 0x)*
    - *octal (8-based - prefixed with 0)*
  - *The PHP **var_dump()** function returns the data type and value.*

```
<html>
<body>
<?php
$x = 1500;
var_dump($x);
?>
</body>
</html>

OUTPUT: int(1500)
```

PHP Programming

PHP Data Types

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

- In the following example $x is a float.
- The PHP var_dump() function returns the data type and value

```
<html>
<body>
<?php
$x = 10.365;
$y = false;
var_dump($x);
var_dump($y);
?>
</body>
</html>

OUTPUT: float(10.365)
        bool(false)
```

PHP Boolean: ***A Boolean represents two possible states: TRUE or FALSE.***

- *Booleans are often used in conditional testing. .*

```
$x = true;
$y = false;
```

PHP Programming

PHP Data Types

***PHP Object:* An object is a data type which stores data and information on how to process that data.**

- *In PHP, an object must be explicitly declared.*
- *First we must declare a class of object.*
- *For this, we use the class keyword.*

```
<html>
<body>
<?php
class Car {
function Car()
{ $this->model = "SUV";
} }
// create an object
$obj = new Car(); // show object properties
echo $obj->model; ?>
</body>
</html>                    OUTPUT: SUV
```

***PHP NULL Value :* Null is a special data type which can have only one value: NULL.**

- *A variable of data type NULL is a variable that has no value assigned to it.*
- *If a variable is created without a value, it is automatically assigned a value of NULL.*

PHP Programming

PHP Data Types

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

- In the following example $x is a float.
- The PHP var_dump() function returns the data type and value

```
<html>
<body>
<?php
$x = 10.365;
var_dump($x);
?>
</body>
</html>

OUTPUT: float(10.365)
```

***PHP Boolean:* A Boolean represents two possible states: TRUE or FALSE.**

- *Booleans are often used in conditional testing. .*

```
$x = true;
$y = false;
```

PHP Programming

Constants in PHP:

A constant is used to store fixed values. We can declare and use constants using define() function.

- ➢ The constant name can have small letters (lower case).
- ➢ Constants once declared is always visible globally.
- ➢ Global variables in a script are visible throughout the script but not inside function.
- ➢ **Syntax:**

  define(name, value, case-insensitive)

- ➢ **Parameters:**
  - ❖ ***name:*** *Specifies the name of the constant*
  - ❖ ***value:*** *Specifies the value of the constant*
  - ❖ ***case-insensitive:*** *Specifies whether the constant name should be case insensitive. Default is false*

Example:

```
define (" gold price",3800, true);
define (" PI",3.14)
```

PHP Programming

Operators in PHP:

Operators are used to perform operations on variables and values.

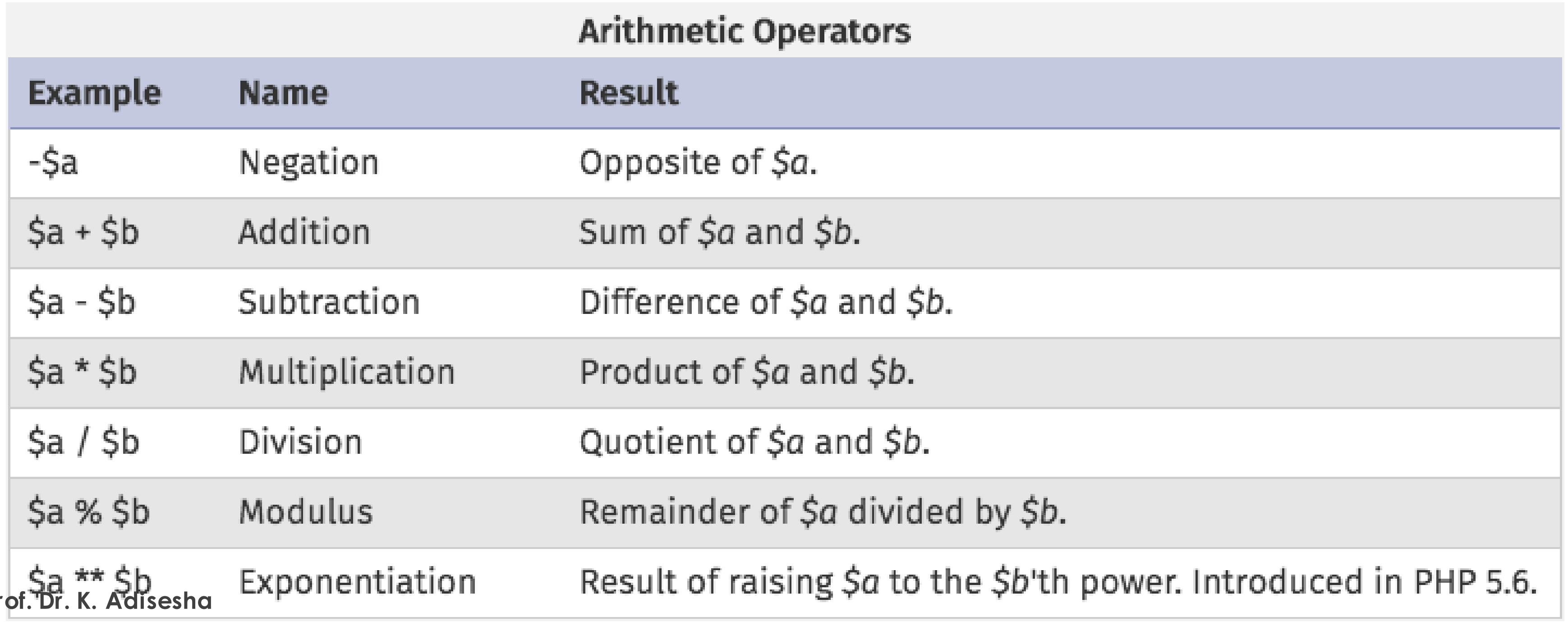
➢ PHP divides the operators in the following groups:

- ❖ *Arithmetic operators*
- ❖ *Assignment operators*
- ❖ *Comparison operators*
- ❖ *Increment/Decrement operators*
- ❖ *Logical operators*
- ❖ *String operators*
- ❖ *Array operators*
- ❖ *Conditional assignment operators*

Operators in PHP

Arithmetic operators in PHP:

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc..



Arithmetic Operators

Example	Name	Result
-$a	Negation	Opposite of $a.
$a + $b	Addition	Sum of $a and $b.
$a - $b	Subtraction	Difference of $a and $b.
$a * $b	Multiplication	Product of $a and $b.
$a / $b	Division	Quotient of $a and $b.
$a % $b	Modulus	Remainder of $a divided by $b.
$a ** $b	Exponentiation	Result of raising $a to the $b'th power. Introduced in PHP 5.6.

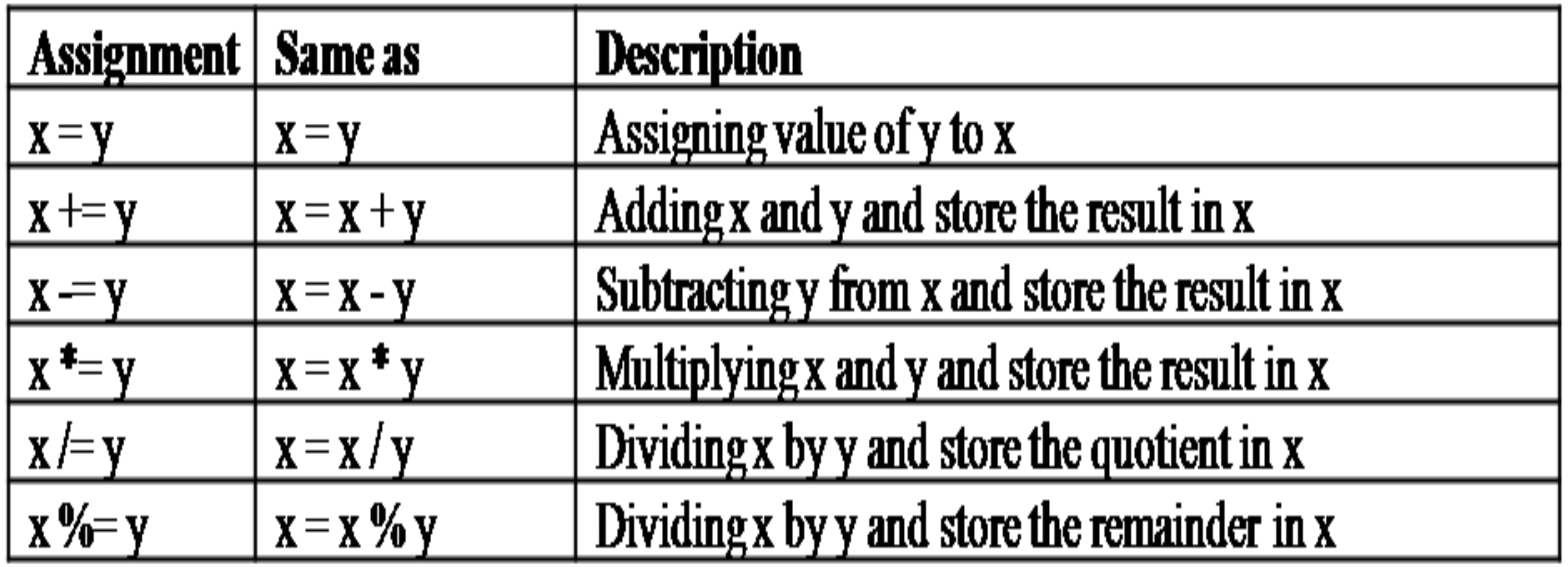
```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 10;
$y = 20;
echo $x + $y;
?>
</body>
</html>
```

Output: 30

Operators in PHP

Assignment operators in PHP:

The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is "=".



Assignment	Same as	Description
x = y	x = y	Assigning value of y to x
x += y	x = x + y	Adding x and y and store the result in x
x -= y	x = x - y	Subtracting y from x and store the result in x
x *= y	x = x * y	Multiplying x and y and store the result in x
x /= y	x = x / y	Dividing x by y and store the quotient in x
x %= y	x = x % y	Dividing x by y and store the remainder in x

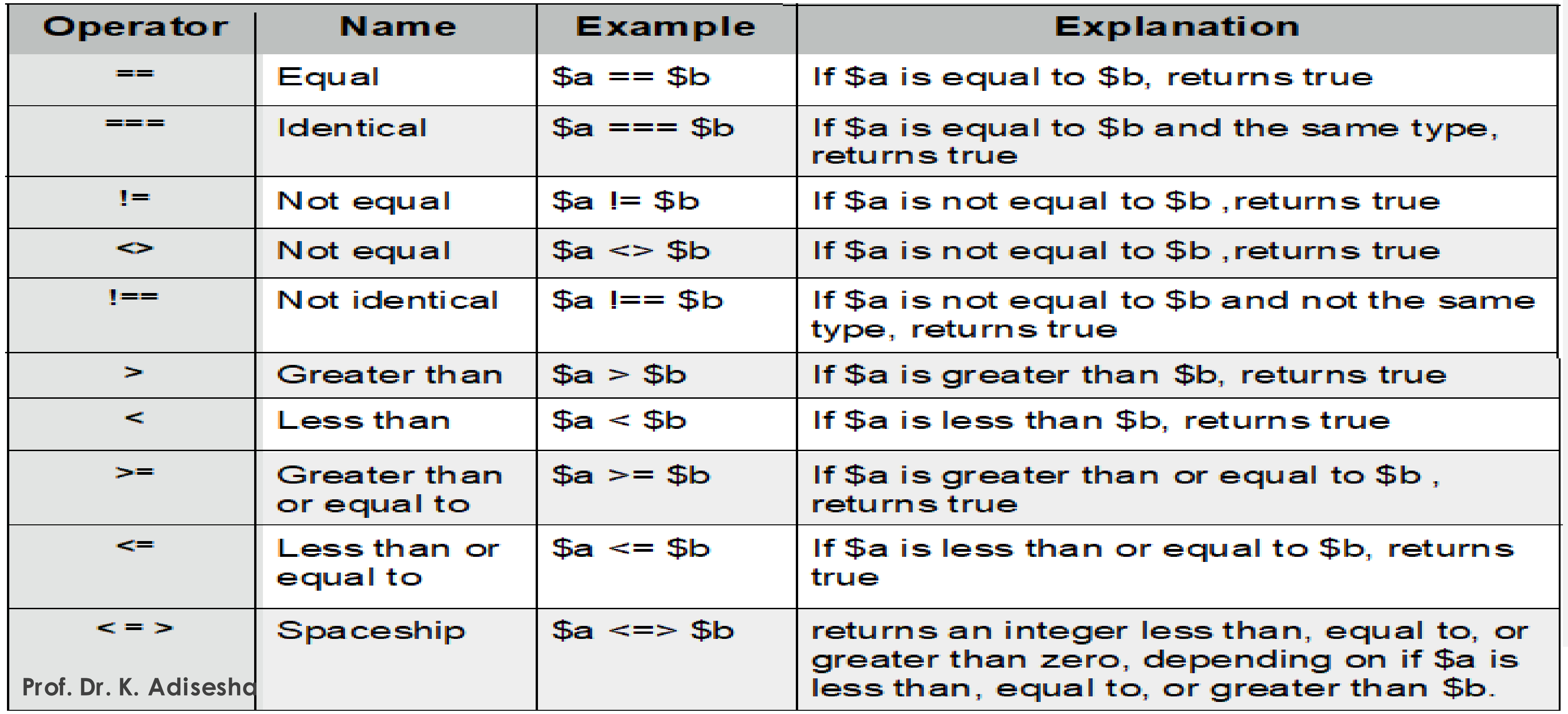
```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 20;
$x += 100;
echo $x;
?>
</body>
</html>
```

Output: 120

Operators in PHP

Comparison operators in PHP:

The PHP comparison operators are used to compare two values (number or string):



Operator	Name	Example	Explanation
==	Equal	$a == $b	If $a is equal to $b, returns true
===	Identical	$a === $b	If $a is equal to $b and the same type, returns true
!=	Not equal	$a != $b	If $a is not equal to $b ,returns true
<>	Not equal	$a <> $b	If $a is not equal to $b ,returns true
!==	Not identical	$a !== $b	If $a is not equal to $b and not the same type, returns true
>	Greater than	$a > $b	If $a is greater than $b, returns true
<	Less than	$a < $b	If $a is less than $b, returns true
>=	Greater than or equal to	$a >= $b	If $a is greater than or equal to $b , returns true
<=	Less than or equal to	$a <= $b	If $a is less than or equal to $b, returns true
< = >	Spaceship	$a <=> $b	returns an integer less than, equal to, or greater than zero, depending on if $a is less than, equal to, or greater than $b.

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 100;
$y = "100";
var_dump($x == $y); // returns
true because values are equal
?>
</body>
</html>
```

Output: bool(true)

Operators in PHP

Increment / Decrement Operators in PHP:

The PHP comparison operators are used to increment / decrement a variable's value.

Operator	Same as...	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

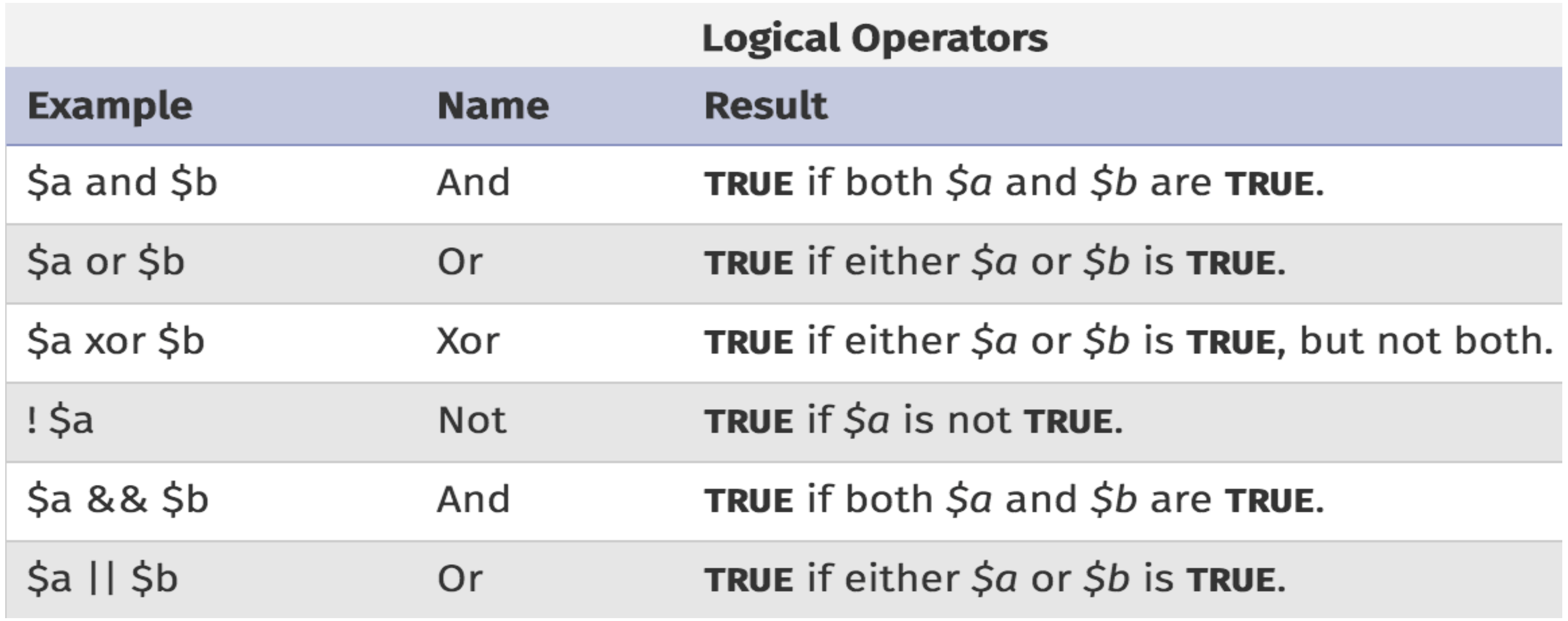
```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 10;
echo ++$x;
?>
</body>
</html>
```

Output: 11

Operators in PHP

Logical operators in PHP:

The PHP logical operators are used to combine conditional statements.



Logical Operators

Example	Name	Result
$a and $b	And	**TRUE** if both *$a* and *$b* are **TRUE**.
$a or $b	Or	**TRUE** if either *$a* or *$b* is **TRUE**.
$a xor $b	Xor	**TRUE** if either *$a* or *$b* is **TRUE**, but not both.
! $a	Not	**TRUE** if *$a* is not **TRUE**.
$a && $b	And	**TRUE** if both *$a* and *$b* are **TRUE**.
$a || $b	Or	**TRUE** if either *$a* or *$b* is **TRUE**.

```
<!DOCTYPE html>
<html>
<body>
<h1>The and Operator</h1>
<?php
$x = 10;
$y = 20;
if ($x == 10 and $y == 20) {
   echo "Welcome to SJES";
}
?>
</body>
</html>
```

Output: The and Operator
Welcome to SJES

Operators in PHP

String operators in PHP:

PHP has two operators that are specially designed for strings.

Operator	Description	Example	Result
.	Concatenation	$str1 . $str2	Concatenation of $str1 and $str2
.=	Concatenation assignment	$str1 .= $str2	Appends the $str2 to the $str1

```
<!DOCTYPE html>
<html>
<body>
<?php
$txt1 = “SJES";
$txt2 = " College!";
echo $txt1 . $txt2;
?>
</body>
</html>
```

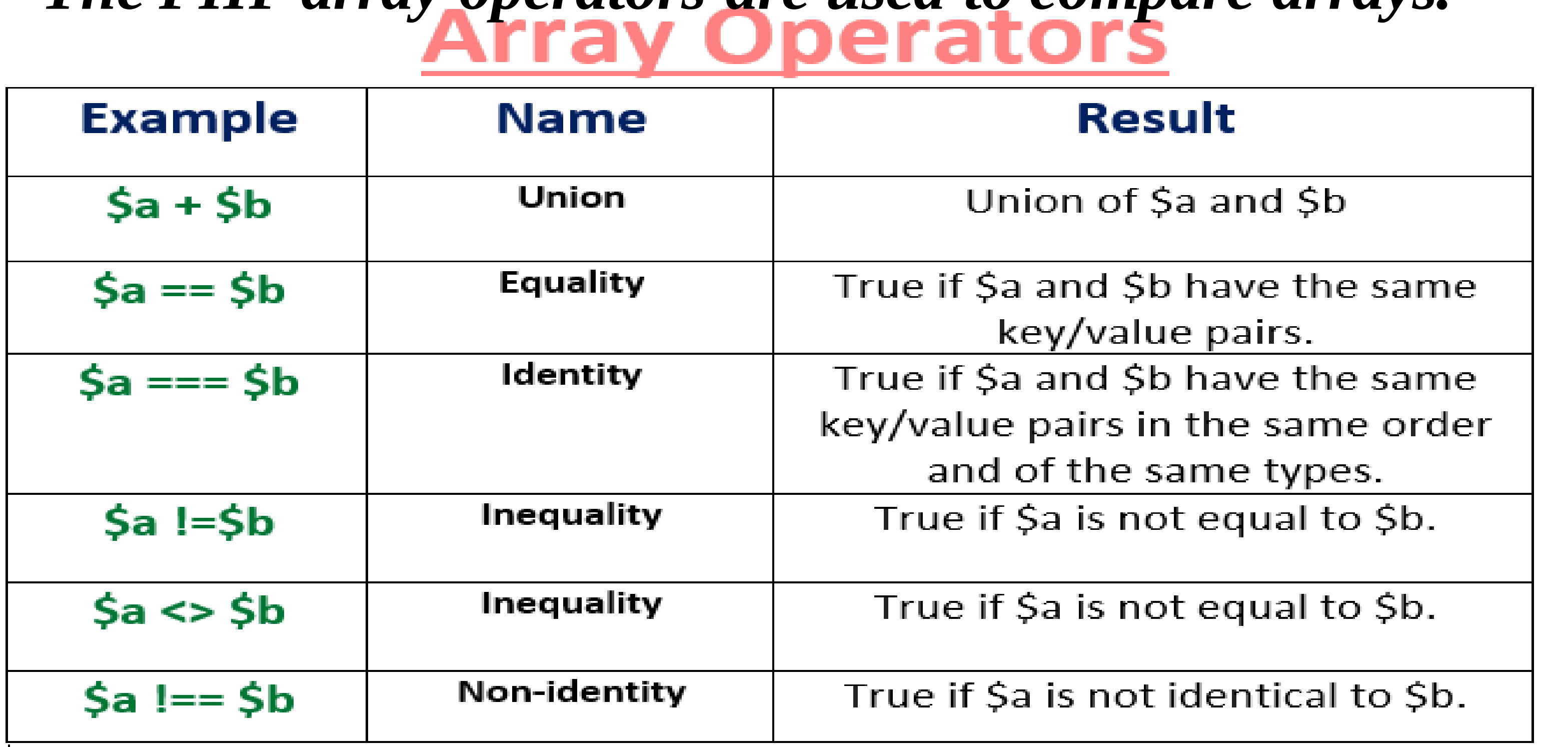
Output:

```
SJES College!
```

Operators in PHP

Array operators in PHP:

The PHP array operators are used to compare arrays.



Array Operators

Example	Name	Result
$a + $b	Union	Union of $a and $b
$a == $b	Equality	True if $a and $b have the same key/value pairs.
$a === $b	Identity	True if $a and $b have the same key/value pairs in the same order and of the same types.
$a !=$b	Inequality	True if $a is not equal to $b.
$a <> $b	Inequality	True if $a is not equal to $b.
$a !== $b	Non-identity	True if $a is not identical to $b.

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = array("a" => “10", "b" => “20");
$y = array("c" => “30", "d" => “40");
print_r($x + $y); // union of $x and $y
?>
</body>
</html>
```

Output:

```
Array ( [a] => 10 [b] => 20 [c] => 30
[d] => 40 )
```

Operators in PHP

Conditional assignment operators:

The PHP conditional assignment operators are used to set a value depending on conditions:

?: ***Ternary***

$x = expr1 ? expr2 : expr3

Returns the value of $x.

The value of $x is expr2 if expr1 = TRUE.

The value of $x is expr3 if expr1 = FALSE

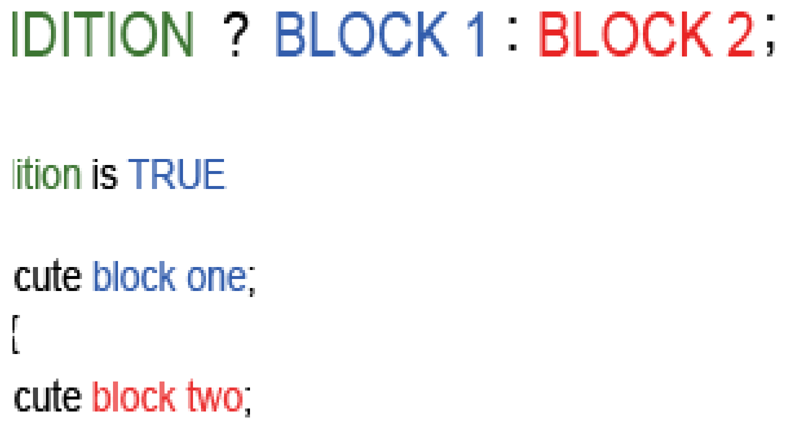
?? ***Null coalescing***

$x = expr1 ?? Expr2

Returns the value of $x.

The value of $x is expr1 if expr1 exists, else NULL.

If expr1 does not exist, or is NULL, the value of $x is expr2.



```
IDITION ? BLOCK 1 : BLOCK 2;

ition is TRUE

cute block one;
[
cute block two;
```

Unit 2 Conditional Statements

Decision Making Statements:

Conditional statements are used to perform different actions based on different conditions.

- In PHP we have the following conditional statements:
  - *if statement - executes some code if one condition is true*
  - *if...else statement - executes some code if a condition is true and another code if that condition is false*
  - *if...elseif...else statement - executes different codes for more than two conditions*
  - *switch statement - selects one of many blocks of code to be execute*

Unit 2 Conditional Statements

Decision Making Statements:

Conditional statements are used to perform different actions based on different conditions.

- *if statement - executes some code if one condition is true*

```
$a = 10;
if ($a == 10) {
  echo “Welcome to SJES College!";
}
```

0r

One-line if statement:

```
$a = 5;
if ($a < 10) $b = "Welcome to SJES College!";
echo $b
```

- *if...else statement - executes some code if a condition is true and another code if that condition is false*

```
if (condition) {
  // code to be executed if condition is true;
} else {
  // code to be executed if condition is
false;
```

Unit 2 Conditional Statements

Decision Making Statements:

Conditional statements are used to perform different actions based on different conditions.

- *if...elseif...else statement*- executes different codes for more than two conditions
  - Example:

```
<?php
$x = "22";
if ($x == "22") {
  echo "correct guess";
} else if ($x < "22") {
  echo "Less than 22";
} else {
  echo "Greater than 22";
}
?>
```

One-line if...else statement:

This technique is known as Ternary Operators

```
$a = 15;
$b = $a < 10 ? "Hello" : "Good Bye";
echo $b;
```

Unit 2 Conditional Statements

Decision Making Statements:

Conditional statements are used to perform different actions based on different conditions.

- *Switch Statement*- This statement allows us to execute different blocks of code based on different conditions. Rather than using if-elseif-if, we can use the switch statement to make our program.
  - *Example:*

```
<?php
$i = "2";
switch ($i) {
    case 0:
        echo "i equals 0";
        break;
    case 1:
        echo "i equals 1";
        break;
    case 2:
        echo "i equals 2";
        break;
    default:
        echo "i is not equal to 0, 1 or 2";
}
?>
```

PHP Iterative Statements

Looping/Iterative Statements:

Iterative statements are used to run same block of code over and over again for a certain number of times.

- In PHP, we have the following loops:
  - ***while -*** *loops through a block of code as long as the specified condition is true.*
  - ***do...while -*** *loops through a block of code once, and then repeats the loop as long as the specified condition is true.*
  - ***for -*** *loops through a block of code a specified number of times.*
  - ***foreach -*** *loops through a block of code for each element in an array.*

PHP Iterative Statements

Looping/Iterative Statements:

Iterative statements are used to run same block of code over and over again for a certain number of times.

- In PHP, we have the following loops:
  - ***while*** **-** *loops through a block of code as long as the specified condition is true.*
  - ***do...while*** **-** *loops through a block of code once, and then repeats the loop as long as the specified condition is true.*
  - ***for*** **-** *loops through a block of code a specified number of times.*
  - ***foreach*** **-** *loops through a block of code for each element in an array.*

PHP Iterative Statements

Looping/Iterative Statements:

while Loop- loops through a block of code as long as the specified condition is true.

- The while loop does not run a specific number of times, but checks after each iteration if the condition is still true.
  - *With the break statement we can stop the loop even if the condition is still true*
  - *With the continue statement we can stop the current iteration, and continue with the next*

```
<?php
$i = 1;
while ($i < 6) {
  if ($i == 3) break;
  echo $i;
  $i++;
}
?>
```

Output: 1 2

OR

```
<?php
$i = 0;
while ($i < 6) {
  $i++;
  if ($i == 3) continue;
  echo $i;
}
?>
```

Output: 1 2 4 5 6

PHP Iterative Statements

Looping/Iterative Statements:

do...while Loop- loops through a block of code once, and then repeats the loop as long as the specified condition is true.

- In a do...while loop the condition is tested AFTER executing the statements within the loop.

```
<!DOCTYPE html>
<html>
<body>
<?php
$i = 10;
do {
  echo $i;
  $i++;
} while ($i < 6);
?>
<p>The code is executed at
least once, if false </p>
</body>
</html>
```

Output: 10

The code is executed at least once, if false

PHP Iterative Statements

Looping/Iterative Statements:

For Loop- loops through a block of code a specified number of times.

- In a do...while loop the condition is tested AFTER executing the statements within the loop.
- **Syntax**

```
for (expression1, expression2, expression3) {
  // code block
}
```

- **This is how it works:**

expression1 is evaluated once
expression2 is evaluated before each iterarion
expression3 is evaluated after each iterarion

```
<!DOCTYPE html>
<html>
<body>
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
</body>
</html>
```

PHP Iterative Statements

Looping/Iterative Statements:

foreach Loop- loops through a block of code for each element in an array.

- The most common use of the foreach loop, is to loop through the items of an array.
- *The array above is an indexed array, where the first item has the key 0, the second has the key 1, and so on.*
- **Example**

```
<!DOCTYPE html>
<html>
<body>
<?php
$course = array("BCA",
"BBA", "B.Com", "BHM");
foreach ($course as $x) {
  echo "$x <br>";
}
?>
</body>
</html>
```

Output:

```
BCA
BBA
B.Com
BHM
```

Arrays

Arrays in PHP:

An array is a collection of data items of the same data type. And it is also known as a subscript variable.

- ➢ PHP array is an ordered map used to hold multiple values of similar type in a single variable.
- ➢ In PHP, there are three different kinds of arrays.:
  - ❖ ***Indexed Array/ Numeric Array -*** *These are arrays with a numeric index where values are stored and accessed in a linear fashion.*
  - ❖ ***Associative Array -****These are arrays with string as an index where it stores element values associated with key values..*
  - ❖ ***Multidimensional Arrays -*** *A multidimensional Array is an array containing one or more arrays where values are accessed using multiple indices.*

Arrays

Indexed Array/ Numeric Array:

These are arrays with a numeric index where values are stored and accessed in a linear fashion.

- PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array.
- ***Example:***

```
<?php
$sub = array(“Java", “Python", “PHP");
echo "I am Studying  " . $sub[0] . ", " . $sub[1] . " and " . $sub[2] . “.";
?>
```

Output:

```
I am Studying Java, Phyton, PHP.
```

Arrays

Associative Array :

These are arrays with string as an index where it stores element values associated with key values.

- Associative arrays are arrays that use named keys that you assign to them..
- **Example:**

```
<!DOCTYPE html>
<html>
<body>
<?php
$salary=array(“Adi"=>"55000",“Sunny"=>“35000","Ram"=>“30000");
echo “Adi salary: ".$salary[“Adi"]."<br/>";
echo "Vimal salary: ".$salary[“Sunny"]."<br/>";
echo "Ratan salary: ".$salary["Ram"]."<br/>";
?> </body>
</html>
```

Output:

```
Adi salary: 55000
Sunny salary: 35000
Ram salary: 30000
```

Arrays

Multidimensional Arrays :

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array.

- PHP multidimensional array can be represented in the form of matrix which is represented by row * column...
- **Example:**

```
<?php
$emp = array
  (
  array(1,“Adi",400000),
  array(2,“Sunny",500000),
  array(3,“Ram",300000)
  );
```

```
for ($row = 0; $row < 3; $row++) {
 for ($col = 0; $col < 3; $col++) {
   echo $emp[$row][$col]."  ";
 }
 echo "<br/>";
}
?>
```

Output:

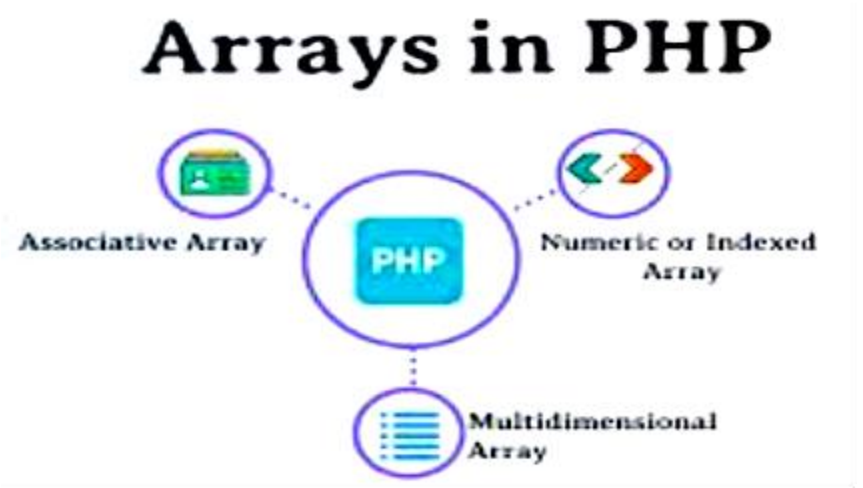
```
1    Adi      55000
2    Sunny    35000
3    Ram      30000
```

Arrays

Working With Arrays:

Array items can be of any data type. The most common are strings and numbers (int, float), but array items can also be objects, functions or even arrays.

- *PHP provides various array functions to access and manipulate the elements of array.*
  - *Create Arrays*
  - *Access Arrays*
  - *Update Arrays*
  - *Add Array Items*
  - *Remove Array Items*
  - *Sort Arrays*



Arrays

PHP Array Functions:

PHP provides various built in array functions to access and manipulate the elements of array.

- ***array()*** *function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.*
- ***array_change_key_case()*** *function changes the case of all key of an array.*
- ***array_chunk()*** *function splits array into chunks. We can divide array into many parts.*
- ***count()*** *function counts all elements in an array.*
- ***sort()*** *function sorts all the elements in an array.*
- ***array_reverse()*** *function returns an array containing elements in reversed order.*
- ***array_search()*** *function searches the specified value in an array.*
- ***array_intersect()*** *function returns the intersection of two array.*

Arrays

Working With Arrays:

PHP provides various array functions to access and manipulate the elements of array.

- *Create Arrays:* You can create arrays by using the array() function:

  Example : *$Course = array("BCA", "BBA", "BHM");*

- *Access Arrays:* To access an array item, you can refer to the index number for indexed arrays, and the key name for associative arrays

  Example: *$Course = array("BCA", "BBA", "BHM");*
  echo $Course[2];

- *Update Arrays:* To update an existing array item, you can refer to the index number for indexed arrays, and the key name for associative arrays.

  Example: *Change the second array item from "BBA" to “BCOM":*
  $Course = array("BCA", "BBA", "BHM");
  $Course[1] = “BCOM";

Arrays

Working With Arrays:

PHP provides various array functions to access and manipulate the elements of array.

- ***Add Array Items:*** To add items to an existing array, you can use the bracket [] syntax.

  Example: *Add one more item to the fruits array:*

  ```
  $fruits = array("Apple", "Banana", "Cherry");
  $fruits[] = "Orange";
  ```

- ***Remove Array Items:*** To remove an existing item from an array, you can use the array_splice() function.

  Example: *Remove the second item:*

  ```
  $Course = array("BCA", "BBA", "BHM");
  array_splice($Course, 1, 1);
  ```

Arrays

Working With Arrays:

PHP provides various array functions to access and manipulate the elements of array.

- Sort Arrays: The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.
- Following are PHP array sort functions:
  - *sort() - sort arrays in ascending order*
  - *rsort() - sort arrays in descending order*
  - *asort() - sort associative arrays in ascending order, according to the value*
  - *ksort() - sort associative arrays in ascending order, according to the key*
  - *arsort() - sort associative arrays in descending order, according to the value*
  - *krsort() - sort associative arrays in descending order, according to the key*

Example:

```
$Course = array("BCA", "BBA", "BHM");
sort($Course);
```

Discussion

Queries ?

Prof. K. Adisesha

9449081542

Thank you