We innovate. We build. We serve.

Our Mission

The University shall primarily provide advanced instruction and professional training in the arts, philosophy, social sciences, agriculture and fishery, forestry, science and technology engineering, education, law and other related fields. It shall also undertake research and extension services and provide progressive leadership in its areas of specialization.

Section 2, R.A. 9313

Our Vison

A transformative University committed to technology, innovation, and service excellence

Board Resolution No. 41, s. 2014

Bachelor Of Science In Information Technology

WD 101 – WEB DEVELOPMENT & APPLICATION

Sonny Boy R. Ellema, Jr.

College of Arts & Sciences

UNIT 5
Server Side Programming- Basic PHP

5.0 Learning Outcomes

After completing this module, you will be able to:

a. Describe what is PHP
b. Identify the steps on how to install PHP
c. Know the basic variables, data types and its rules
d. Display content using Echo/Print functions
e. Describe the different operators in PHP
f. What are the conditional statements in PHP
g. Describe the different type of loops in PHP

5.1 Introduction

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications. This tutorial helps you to build your base with PHP.

5.2. How to use PHP Why to Learn PHP?

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994. **PHP** is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning PHP:

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.

- PHP Syntax is C-Like.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Hello World using PHP.

Just to give you a little excitement about PHP, I'm going to give you a small conventional PHP Hello World program, You can try it using Demo link.

```
<html>

  <head>
    <title>Hello World</title>
  </head>

  <body>
    <?php echo "Hello, World!";?>
  </body>

</html>
```

Applications of PHP

As mentioned before, PHP is one of the most widely used language over the web. I'm going to list few of them here:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Audience

This **PHP tutorial** is designed for PHP programmers who are completely unaware of PHP concepts but they have basic understanding on computer programming.

Prerequisites

Before proceeding with this tutorial you should have at least basic understanding of computer programming, Internet, Database, and MySQL etc is very helpful.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible. PHP Syntax is C-Like.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website. It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this −

```
<html>

  <head>
    <title>Hello World</title>
  </head>

  <body>
    <?php echo "Hello, World!";?>
  </body>

</html>
```

It will produce following result − Hello,
World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

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

<?   PHP code goes here ?>

<script language = "php"> PHP code goes here </script>
```

A most common tag is the <?php...?> and we will also use the same tag in our tutorial. From the next chapter we will start with PHP Environment Setup on your machine and then we will dig out almost all concepts related to PHP to make you comfortable with the PHP language.

5.3. How to install PHP?

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

- **Web Server** − PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here − https://httpd.apache.org/download.cgi
- **Database** − PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here − https://www.mysql.com/downloads/

- **PHP Parser** − In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

PHP Parser Installation

Before you proceed it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP.
Type the following address into your browser's address box.
http://127.0.0.1/info.php
If this displays a page showing your PHP installation related information then it means you have PHP and Webserver installed properly. Otherwise you have to follow given procedure to install PHP on your computer.
This section will guide you to install and configure PHP over the following four platforms −

- PHP Installation on Linux or Unix with Apache
- PHP Installation on Mac OS X with Apache
- PHP Installation on Windows NT/2000/XP with IIS
- PHP Installation on Windows NT/2000/XP with Apache

Apache Configuration

If you are using Apache as a Web Server then this section will guide you to edit Apache Configuration Files.
Just Check it here − PHP Configuration in Apache Server

PHP.INI File Configuration

The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality.
Just Check it here − PHP.INI File Configuration

Windows IIS Configuration

To configure IIS on your Windows machine you can refer your IIS Reference Manual shipped along with IIS.

5.4. PHP Syntax

This chapter will give you an idea of very basic syntax of PHP and very important to make your PHP foundation strong.

Escaping to PHP

The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP'. There are four ways to do this −

Canonical PHP tags

The most universally effective PHP tag style is −
<?php...?>

If you use this style, you can be positive that your tags will always be correctly interpreted.
Short-open (SGML-style) tags
Short or short-open tags look like this − <?...?>
Short tags are, as one might expect, the shortest option You must do one of two things to enable PHP to recognize the tags −

- Choose the --enable-short-tags configuration option when you're building PHP.
- Set the short_open_tag setting in your php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags. **ASP-style tags**

ASP-style tags mimic the tags used by Active Server Pages to delineate code blocks. ASPstyle tags look like this −
<%...%>
To use ASP-style tags, you will need to set the configuration option in your php.ini file. **HTML script tags**
HTML script tags look like this − <script
language = "PHP">...</script>

Commenting PHP Code

A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP − **Single-line comments** − They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

```
<?
  # This is a comment, and
  # This is the second line of the comment

  // This is a comment too. Each style comments only    print "An example
with single line comments";
?>
```

Multi-lines printing − Here are the examples to print multiple lines in a single print statement −

```
<?
  #     First     Example
print <<<END
  This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note   that
the here document terminator must appear on a    line
with just a semicolon no extra whitespace!
  END;

  # Second Example
print "This spans
  multiple   lines.   The   newlines   will   be
output as well";
?>
```

Multi-lines comments − They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here are the example of multi lines comments.

```
<?
 /* This is a comment with multiline
   Author : Mohammad Mohtashim
   Purpose: Multiline Comments Demo
   Subject: PHP
 */

 print "An example with multi line comments"; ?>
```

PHP is whitespace insensitive

Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).
PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row.one whitespace character is the same as many such characters.
For example, each of the following PHP statements that assigns the sum of 2 + 2 to the variable $four is equivalent −

```
$four = 2 + 2; // single spaces
$four <tab>=<tab2<tab>+<tab>2 ; // spaces and tabs
$four =
2+
2; // multiple lines
```

PHP is case sensitive

Yeah it is true that PHP is a case sensitive language. Try out following example −

```
<html>
 <body>

   <?php
    $capital = 67;        print("Variable capital is $capital<br>");
    print("Variable CaPiTaL is $CaPiTaL<br>");
   ?>

 </body>
</html>
```

This will produce the following result − Variable
capital is 67
Variable CaPiTaL is

Statements are expressions terminated by semicolons

A *statement* in PHP is any expression that is followed by a semicolon (;).Any sequence of valid PHP statements that is enclosed by the PHP tags is a valid PHP program. Here is a typical statement in PHP, which in this case assigns a string of characters to a variable called $greeting −

```
$greeting = "Welcome to PHP!";
```

Expressions are combinations of tokens

The smallest building blocks of PHP are the indivisible tokens, such as numbers (3.14159), strings (.two.), variables ($two), constants (TRUE), and the special words that make up the syntax of PHP itself like if, else, while, for and so forth

Braces make blocks

Although statements cannot be combined like expressions, you can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.

Here both statements are equivalent −

```
if (3 == 2 + 1)
  print("Good - I haven't totally lost my mind.<br>");

if (3 == 2 + 1) {
  print("Good - I haven't totally");
  print("lost my mind.<br>");
}
```

Running PHP Script from Command Prompt

Yes you can run your PHP script on your command prompt. Assuming you have following content in test.php file

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

Now run this script as command prompt as follows − $
php test.php

It will produce the following result − Hello
PHP!!!!!

Hope now you have basic knowledge of PHP Syntax.

5.5. PHP Variables & Data Types

The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.

- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables −
Integers − are whole numbers, without a decimal point, like 4195.

- **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** − have only two possible values either true or false.
- **NULL** − is a special type that only has one value: NULL.
- **Strings** − are sequences of characters, like 'PHP supports string operations.' **Arrays** − are named and indexed collections of other values.
- **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.
We will explain only simple data type in this chapters. Array and Objects will be explained separately.

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so − $int_var = 12345;
$another_int = -12345 + 12345;
Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.
For most common platforms, the largest integer is (2**31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2**31 . 1) (or .2,147,483,647).

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code −

```
<?php
  $many = 2.2888800;
  $many_2 = 2.2111200;
  $few = $many + $many_2;

  print("$many + $many_2 = $few <br>");
?>
```

It produces the following browser output − 2.28888
+ 2.21112 = 4.5

Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −

```
if (TRUE)
  print("This will always print<br>");

else   print("This will never print<br>");
```

Interpreting other types as Booleans

Here are the rules for determine the "truth" of any value not already of the Boolean type −
If the value is a number, it is false if exactly equal to zero and true otherwise.

- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;
$true_str = "Tried and true"
$true_array[49] = "An array element";
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999; $false_str = "";
```

NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this −

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed −

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties −
It evaluates to FALSE in a Boolean context.
 It returns FALSE when tested with IsSet() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
  $variable = "name";
  $literally = 'My $variable will not print!';

  print($literally);
  print "<br>";

  $literally = "My $variable will print!";
  print($literally);
?>
```

This will produce following result − My
$variable will not print!
My name will print

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are −

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Here Document

You can assign multiple lines to a single string variable using here document −

```
<?php
  $channel =<<<_XML_

  <channel>
```

```
    <title>What's For Dinner</title>
    <link>http://menu.example.com/ </link>
    <description>Choose what to eat tonight.</description>
  </channel>
  _XML_;

  echo <<<END
  This uses the "here document" syntax to output multiple lines with variable
interpolation. Note that the here document terminator must appear on a line with     just
a semicolon. no extra whitespace!


  END;

  print $channel; ?>
```

This will produce following result −

```
This uses the "here document" syntax to output
multiple lines with variable interpolation. Note that
the here document terminator must appear on a line
with just a semicolon. no extra whitespace!

<channel>
<title>What's For Dinner<title>
<link>http://menu.example.com/<link>
<description>Choose what to eat tonight.</description>
```

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types − ▯ Local variables

- Function parameters
- Global variables
- Static variables

Variable Naming

Rules for naming a variable is −

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc There is no size limit for variables.

5.6. PHP Echo/Print

PHP - Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well. There are two parts which should be clear to you −

- Creating a PHP Function
- Calling a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Please refer to PHP Function Reference for a complete set of useful functions.

Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it. Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below −

```
<html>

  <head>
    <title>Writing PHP Function</title>
  </head>


  <body>

    <?php
      /*   Defining   a   PHP   Function   */
function writeMessage() {
        echo "You are really a nice person, Have a nice time!";
      }

      /* Calling a PHP Function */
      writeMessage();
    ?>

  </body>
</html>
```

This will display following result −
You are really a nice person, Have a nice time!

PHP Functions with Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

```
<html>

  <head>
    <title>Writing PHP Function with Parameters</title>
  </head>

  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
      }

      addFunction(10, 20);
    ?>

  </body>
</html>
```

This will display following result −
Sum of the two numbers is : 30

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.
Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition. Following example depicts both the cases.

```
<html>

  <head>
    <title>Passing Argument by Reference</title>
  </head>
  <body>

    <?php
      function addFive($num) {
        $num += 5;
      }

      function addSix(&$num) {
        $num += 6;
      }

      $orignum = 10;
      addFive( $orignum );

      echo "Original Value is $orignum<br />";

      addSix( $orignum );
      echo "Original Value is $orignum<br />";
    ?>

  </body>
</html>
```

This will display following result − Original
Value is 10
Original Value is 16

PHP Functions returning value

A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.
You can return more than one value from a function using **return array(1,2,3,4)**.
Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that **return** keyword is used to return a value from a function.

```
<html>
```

```
  <head>
    <title>Writing PHP Function which returns value</title>
  </head>
  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        return $sum;
      }
      $return_value = addFunction(10, 20);

      echo "Returned value from the function : $return_value";
    ?>

  </body>
</html>
```

This will display following result −
Returned value from the function : 30

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it. Following function prints NULL in case use does not pass any value to this function.

```
<html>

  <head>
    <title>Writing PHP Function which returns value</title>
</head>

  <body>

    <?php
      function printMe($param = NULL) {
print $param;
      }

      printMe("This is test");
printMe();
    ?>

  </body>
</html>
```

This will produce following result − This
is test

Dynamic Function Calls

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behaviour.

```
<html>

  <head>
    <title>Dynamic Function Calls</title>
  </head>

  <body>

    <?php        function
sayHello() {
        echo "Hello<br />";
      }

      $function_holder = "sayHello";
      $function_holder();
    ?>

  </body>
</html>
```

This will display following result −

```
Hello
```

5.7. PHP Operators

What is Operator? Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language −

Assume variable A holds 10 and variable B holds 20 then −

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators

There are following comparison operators supported by PHP language
Assume variable A holds 10 and variable B holds 20 then −

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

Logical Operators

There are following logical operators supported by PHP language
Assume variable A holds 10 and variable B holds 20 then −

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

There are following assignment operators supported by PHP language −

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
-=	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

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax −

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Operators Categories

All the operators we have discussed above can be categorised into following categories −
Unary prefix operators, which precede a single operand.

- Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
- The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Assignment operators, which assign a value to a variable.

Precedence of PHP Operators

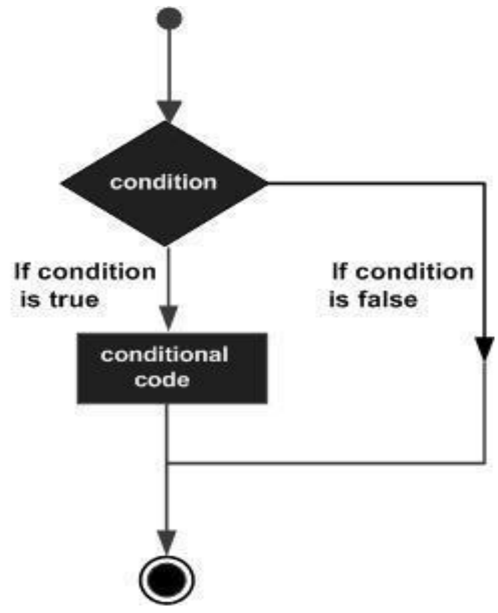
Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator − For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.
Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right

Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	| |	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

5.8. PHP If and Else Statements

The if, elseif ...else and switch statements are used to take decision based on the different condition.
You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements –



- **if...else statement** − use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** − is used with the if...else statement to execute a set of code if **one** of the several condition is true

- **switch statement** − is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax :

if (*condition*)
  code to be executed if condition is true; else
  code to be executed if condition is false;

Example

The following example will output "Have a nice weekend!" if the current day is Friday, Otherwise, it will output "Have a nice day!":

```
<html>
  <body>

    <?php
      $d = date("D");

      if ($d == "Fri")
        echo "Have a nice weekend!";

else
        echo "Have a nice day!";
    ?>

  </body>
</html>
```

It will produce the following result − Have
a nice weekend!

The ElseIf Statement

If you want to execute some code if one of the several conditions are true use the elseif statement.

Syntax :

if (*condition*)
  code to be executed if condition is true;
elseif (*condition*)
  code to be executed if condition is true; else
  code to be executed if condition is false;

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise, it will output "Have a nice day!" −

```
<html>
  <body>

    <?php
      $d = date("D");

      if ($d == "Fri")
        echo "Have a nice weekend!";

      elseif ($d == "Sun")
        echo "Have a nice Sunday!";

else
        echo "Have a nice day!";
    ?>

  </body>
</html>
```

It will produce the following result − Have a nice Weekend!

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax :

switch (*expression*){ case *label1:* *code to be executed if expression = label1;* break;

case *label2:* *code to be executed if expression = label2;* break; default:

code to be executed if expression is different from both label1 and label2;
}

Example

The *switch* statement works in an unusual way. First it evaluates given expression then seeks a lable to match the resulting value. If a matching value is found then the code associated with the matching label will be executed or if none of the lable matches then statement will execute any specified default code.

```
<html>
  <body>
    <?php
      $d = date("D");

      switch ($d){
        case "Mon":
          echo "Today is Monday";
          break;
        case "Tue":
          echo "Today is Tuesday";
          break;
        case "Wed":
          echo "Today is Wednesday";
          break;
        case "Thu":
          echo "Today is Thursday";
          break;
        case "Fri":
          echo "Today is Friday";
          break;
        case "Sat":
          echo "Today is Saturday";
break;
        case "Sun":
          echo "Today is Sunday";
break;
        default:
          echo "Wonder which day is this ?";
      }
    ?>

  </body>
</html>
```

It will produce the following result − Today
is Monday

5.9. PHP Loops

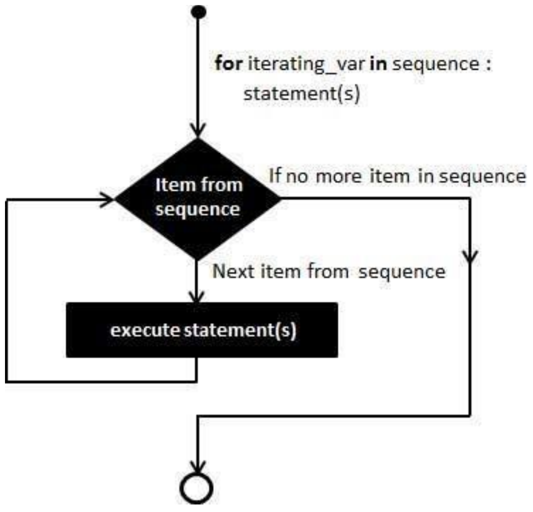
Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

- **for** − loops through a block of code a specified number of times.
- **while** − loops through a block of code if and as long as a specified condition is true.
- **do...while** − loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** − loops through a block of code for each element in an array.

We will discuss about **continue** and **break** keywords used to control the loops execution.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.



Syntax :

for (*initialization*; *condition*; *increment*){
code to be executed;
}

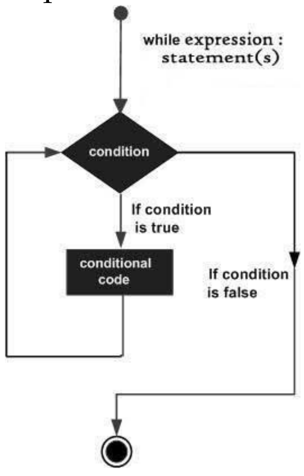
The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i. **Example**
The following example makes five iterations and changes the assigned value of two variables on each pass of the loop −

```
<html>
  <body>

    <?php
      $a = 0;
      $b = 0;

      for( $i = 0; $i<5; $i++ ) {
        $a += 10;
        $b += 5;
      }

      echo ("At the end of the loop a = $a and b = $b" );
    ?>

  </body>
</html>
```

This will produce the following result −
At the end of the loop a = 50 and b = 25

The while loop statement

The while statement will execute a block of code if and as long as a test expression is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.



Syntax :
while (*condition*) {
  code to be executed;
}

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

```
<html>
  <body>

    <?php
      $i = 0;
      $num = 50;

      while( $i < 10) {
        $num--;
        $i++;
      }

      echo ("Loop stopped at i = $i and num = $num" );
    ?>

  </body>
</html>
```

This will produce the following result −

```
Loop stopped at i = 10 and num = 40
```

The do...while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax :

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

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 −

```
<html>
  <body>

    <?php
      $i = 0;
      $num = 0;

do {
        $i++;
      }

      while( $i < 10 );
      echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

This will produce the following result −

```
Loop stopped at i = 10
```

The foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax :

```
foreach (array as value) {
   code to be executed;
}
```

Example

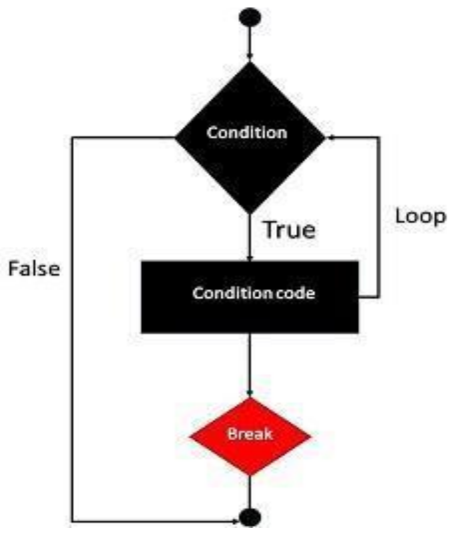
Try out following example to list out the values of an array.

```
<html>
  <body>

    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {
        echo "Value is $value <br />";
      }
    ?>

  </body>
</html>
```

This will produce the following result − Value
is 1
Value is 2
Value is 3
Value is 4
Value is 5

The break statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely. The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.



Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

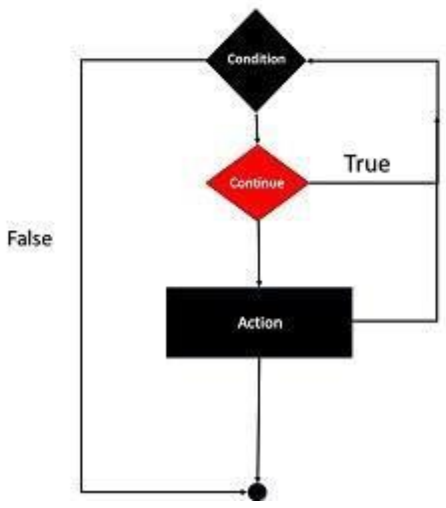
```
<html>
  <body>

    <?php
      $i = 0;

      while( $i < 10) {
        $i++;
        if( $i == 3 )break;
      }
      echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

This will produce the following result − Loop
stopped at i = 3

The continue statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.

Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.



Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
  <body>

    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {           if( $value == 3 )continue;
        echo "Value is $value <br />";
      }
    ?>

  </body>
</html>
```

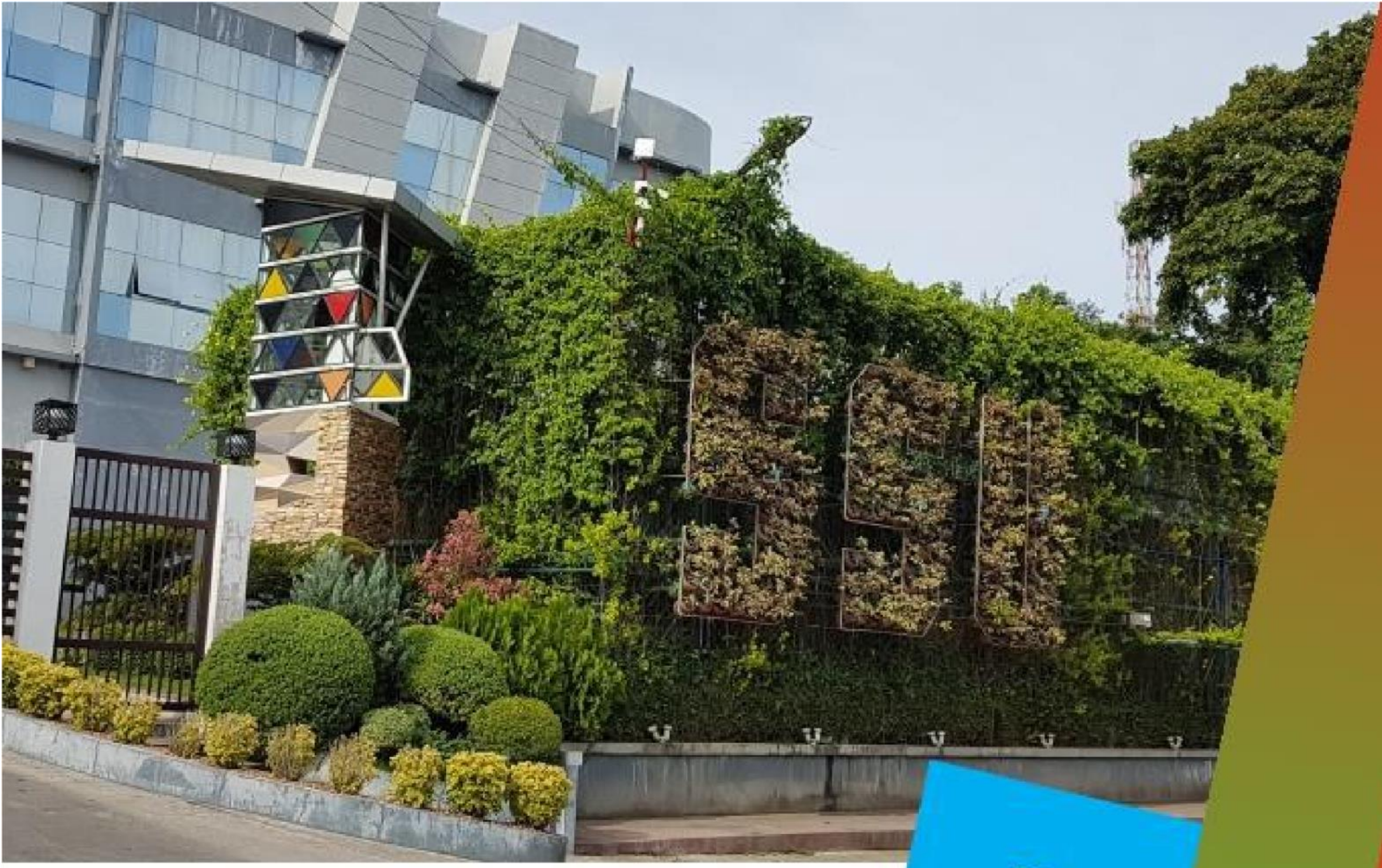
This will produce the following result − Value
is 1
Value is 2
Value is 4
Value is 5

5.10 References

W3Schools HTML5 (http://www.w3schools.com/)
W3Schools CSS3 (http://www.w3schools.com/)
W3Schools PHP (http://www.w3schools.com/)
Ian Lunn. CSS3 Foundations, A John Wiley and Sons, Ltd, Publication 2013
Matt West. HTML5 Foundations, A John Wiley and Sons, Ltd, Publication 2013
Learning web design : a beginner's guide to HTML, CSS, Javascript, and web graphics by Niederst Robbins, Jennifer, Ltd, Publication 2014
Tutorialspoint (https://www.tutorialspoint.com/index.htm)

5.11 Acknowledgement

All the figures and information presented in this module were taken from the references enumerated above.



SAMAR STATE UNIVERSITY
ARTECHE BOULEVARD, CATBALOGAN CITY
6700 SAMAR

www.ssu.edu.ph

email: info@ssu.edu.ph
telefax: (055) 543-8394