UNIT II: Programming with PHP

2.1 Creating an HTML Form,
2.2 Handling an HTML Form
2.3 Managing Magic Quotes
2.4 Conditionals and Operators
2.5 Validating Form Data
2.6 What Are Arrays?
2.7 For and While Loops

2.1 Creating an HTML Form

For creating HTML form, HTML provides <form> tag and different tag for creating different controls. HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called controls like input box, checkboxes, radio-buttons, submit buttons, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for processing.

The <form> tag is used to create an HTML form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the type attribute. An input element can be of type text field, checkbox, password field, radio button, submit button, reset button, etc.

The most used input types are described below.

Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an <input> element, whose type attribute has a value of text. Here's an example of a single-line text input used to take username:

Example

```
<form>
Username:<input type="text" name="username" id="username">
</form>
```

Username:

Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked i.e. shown as asterisks or dots. This is to prevent others from reading the password on the screen. This is also a single-line text input controls created using an <input> element whose type attribute has a value of password.

Example

```
<form>
Password:<input type="password" name="user-password">
</form>
```

Password:

Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of radio.

Example

```
<form>
<input type="radio" name="sex" id="male">Male
<input type="radio" name="sex" id="female">Female
</form>
```

Male Female

Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of checkbox.

Example

```
<form>
<input type="checkbox" name="soccer">Soccer
<input type="checkbox" name="cricket">Cricket
<input type="checkbox" name="baseball">Baseball
</form>
```

Soccer Cricket Baseball

File Select box

The file fields allow a user to browse for a local file and send it as an attachment to the form data. It normally rendered as a text box with a button that enables the user to browse for a file. However, the user can also type the path and name of the file in the text box.

This is also created using an <input> element, whose type attribute value is set to file.

Example

```
<form>
Upload:<input type="file" name="upload">
</form>
```

Upload: Choose File No file chosen

Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an <textarea>

element.

Example

```
<form>
Address<textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

Address:

Select Boxes

A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the <select> element and <option> element. The option elements within the <select> element define each list item.

Example

```
<form>
City:
<select name="city" id="city">
<option>Sydney</option>
<option>Pune</option>
<option>Mumbai</option>
</select>
</form>
```

City: Sydney

Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data. A reset button resets all the forms control to default values.

Example

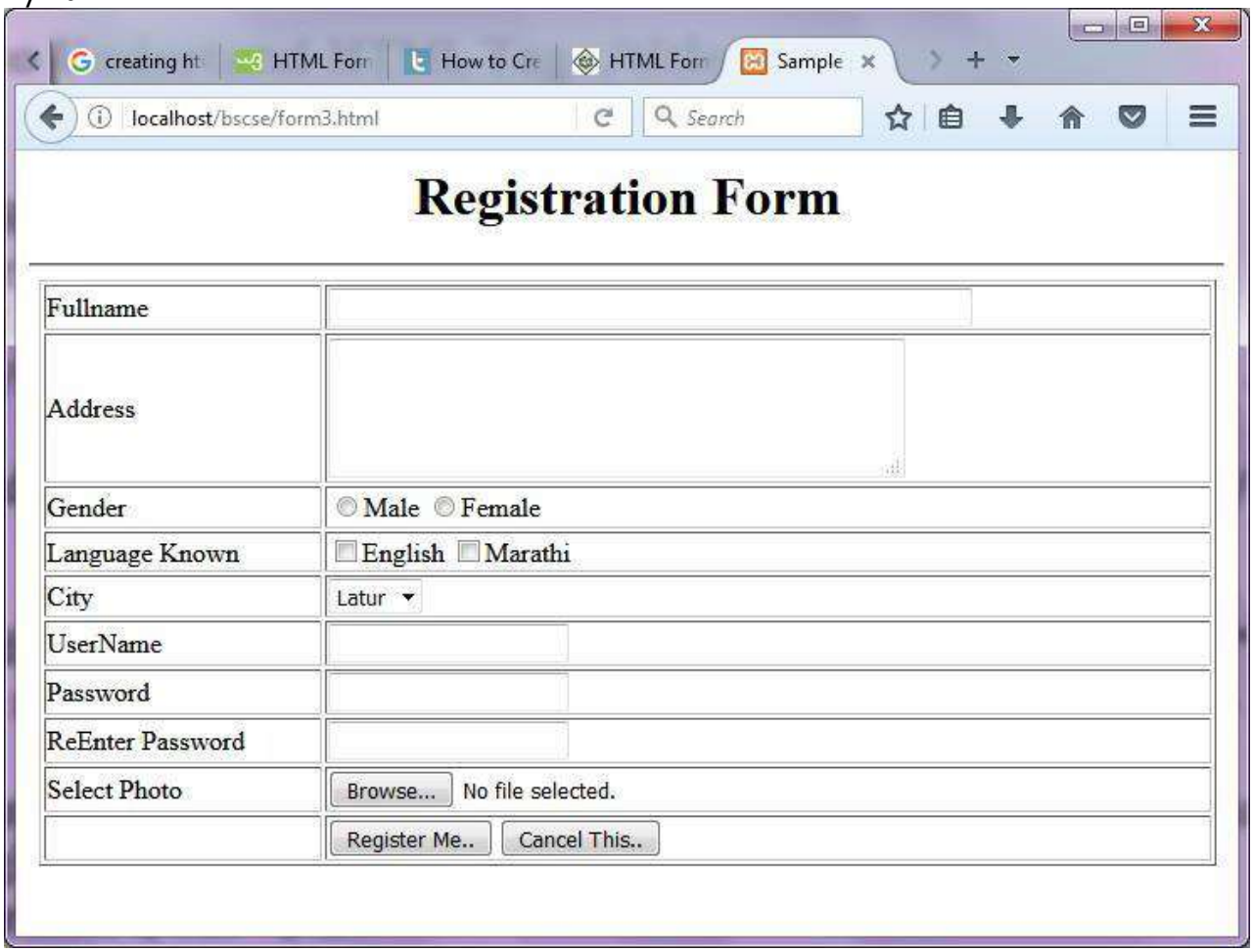
```
<form action="read.php" method="post" First Name:
<input type="text" name="first-name">
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>
```

Example: Creating an HTML Form

```
<html>
<head>
<title>Sample Page to Undertand HTML Form Tag</title>
</head>
```

```
<body>
<h1 align="center">Registration Form</h1>
<hr>
<form method="post">
<table border="1" width="700" align="center">
<tr>
<td>Fullname </td>
<td><input type="text" name="t1" size="60"> </td>
</tr>
<tr>
<td>Address</td>
<td><textarea rows="4" cols="40" name="t2"></textarea></td>
</tr>
<tr>
<td>Gender </td>
<td>
<input type="radio" name="gender" value="Male">Male
<input type="radio" name="gender" value="Female">Female
</td>
</tr>
<tr><td>Language Known </td>
<td>
<input type="checkbox" name="lang1" value="English">English
<input type="checkbox" name="lang2" value="Marathi">Marathi
</td>
</tr>
<tr>
<td>City</td>
<td>
<select><option>Latur</option><option>Pune</option></select>
</td>
</tr>
<tr>
<td>UserName </td>
<td><input type="text" name="t3"></td>
</tr>
<tr>
<td>Password </td>
<td><input type="password" name="t4"></td>
</tr>
<tr>
<td>ReEnter Password </td>
<td><input type="password" name="t5"></td>
```

```
</tr>
<tr>
<td>Select Photo </td>
<td><input type="file" name="f1"></td>
</tr>
<tr>
<td> </td>
<td>
<input type="submit" name="b1" value="Register Me..">
<input type="Reset" name="b1" value="Cancel This..">
</td>
</tr>
</table>
</form>
</body>
</html>
```



2.2 Handling an HTML Form

Handling HTML form in PHP is very simple, when user submit the form, all data is stored and is available in array called $_GET or $_POST depending upon the form method.

$_GET

Whenever user submits the form whose method is GET , then all the information entered on form will be stored in the array $_GET by PHP.
$_GET array contains all parameters or field values that are part of a GET request, where the keys or indexes of the array are the names of the form parameters.

Syntax

$variable_name=$_GET [“form field name”];
Ex: $name=$_GET [“t1”];
Where “t1” is name of text box filed in a form

$_POST

Whenever user submits the form whose method is POST, then all the information entered on form will be stored in the array $_POST by PHP.
$_POST array contains all parameters that are part of a POST request, where the keys of the array are the names of the form parameters

Syntax

$variable_name=$_POST [“form field name”];
Ex: $name=$_POST [“fullname”];
Where “fullname” is name of text box field in a form

$_REQUEST

$_REQUEST is also used for reading or handling form data if the form method is either GET or POST.

Syntax

$variable_name=$_REQUEST [“form field name”];
Ex: $name=$_REQUEST [“gender”];
Where “gender” is name of text box field in a form

Example: form1.html

```
<html>
<head><title>User Info Form</title></head>
<body>
<h1> User Info Form </h1>
```

```
<form action="read.php" method="GET">
Enter the Fullname: <input type="text" name="t1" /><br /> Enter your
age<input type="text" name="t2" /><br />
<input type="submit" value="Submit Data">
</form>
</body>
</html>
```

2.3 Managing Magic Quotes

Magic quotes was a feature of the PHP language before PHP version 6, through which strings are automatically escaped special characters such as single quote(„), double quote(“) are prefixed with a backslash before it is displayed.

Magic Quotes—when enabled—automatically escapes single and double quotation marks found in submitted form data.

You can be also use the **stripslashes**() function.

$var = stripslashes($var);

This function will remove any backslashes found in $var. This will have the effect of turning an escaped submitted string back to its original, non-escaped value.

Example:

magicquote.html

```
<h1>Your Comments </h1><hr>
<form action="magicquote.php">
<textarea rows="6" cols="70" name="t1">
</textarea>
<br>
<input type="submit" value= "Send Data">
</form>
```

magicquote.php

```
<?php
$c=$_REQUEST["t1"];
//$c = stripslashes($c);
print "<h1> Comments given :$c";
?>
```

2.4 Conditionals and Operators

Conditional Statements

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

if
if-else
if-else-if
nested if

If Statement

PHP if statement allows conditional execution of code. It is executed if condition is true.

If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

Syntax

Example

```
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

Output: 12 is less than 100

If-else Statement

PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is true and another block of code if the condition is false.

Syntax

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

Example

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

If-else-if Statement

The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

Syntax

```
if (condition1){
//code to be executed if condition1 is true
} elseif (condition2){
//code to be executed if condition2 is true
} elseif (condition3){
//code to be executed if condition3 is true
....
}  else{
//code to be executed if all given conditions are false
}
```

Example

```
<?php
   $marks=69;
   if ($marks<33){
      echo "fail";
   }
   else if ($marks>=34 && $marks<50) {
      echo "D grade";
   }
   else if ($marks>=50 && $marks<65) {
     echo "C grade";
   }
```

```
   else if ($marks>=65 && $marks<80) {
      echo "B grade";
   }
   else if ($marks>=80 && $marks<90) {
      echo "A grade";
   }
   else if ($marks>=90 && $marks<100) {
      echo "A+ grade";
   }
  else {
      echo "Invalid input";
   }
?>
```

Output: B Grade

Nested if Statement

The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is true.

Syntax

```
if (condition0) {
//code to be executed if condition is true
                if (condition1)
                {
                        //code to be executed if condition1 is true
                }
                else
                {
                        //code to be executed if condition1 is false
                }
}
else           ////code to be executed if condition0 is false
{
                if (condition2)
                {
                        //code to be executed if condition2 is true
                }
                else
                {
                        //code to be executed if condition2 is false
                }
```

```
}
```

Example:

```
<?php
        $age = 23;
  $nationality = "Indian";
  //applying conditions on nationality and age
  if ($nationality == "Indian")
  {
    if ($age >= 18) {
      echo "Eligible to give vote";
    }
    else {
      echo "Not eligible to give vote";
    }
  }
?>
```

Switch statement

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax

```
switch(expression){
case value1:
 //code to be executed
 break;
case value2:
 //code to be executed
 break;
......
default:
 code to be executed if all cases are not matched;
}
```

Important points to be noticed about switch case:

1. The default is an optional statement. Even it is not important, that default must always be the last statement.
2. There can be only one default in a switch statement. More than one default may lead to a Fatal error.

3. Each case can have a break statement, which is used to terminate the sequence of statement.
4. The break statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.
5. PHP allows you to use number, character, string, as well as functions in switch expression.
6. Nesting of switch statements is allowed, but it makes the program more complex and less readable.
7. You can use semicolon (;) instead of colon (:). It will not generate any error.

Example

```
<?php
$num=20;
switch($num){
case 10:
echo("number is equals to 10");
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

Output:
number is equal to 20

2.4.2 Operators

"Operators are used to perform operations on variables or values".

Example:
$num=10+20;

We can categorize operators on behalf of operands. They can be categorized in 3 forms:

Unary Operators: works on single operands such as ++, -- etc.

Binary Operators: works on two operands such as binary +, -, *, / etc.

Ternary Operators: works on three operands such as "?:".

PHP Operators can be categorized in following types:

Arithmetic Operators

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

Ex. +,-,*,/,%,**.

**-Exponentiation - $a raised to the power $b

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "="

Ex =,+=,-=,*=,/=,%=.

Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Ex. & - AND , |-OR , ^-Xor, ~ -Not.

Comparison Operators

Comparison operators allow comparing two values, such as number or string.

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or Equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal Return 1 if $a is greater than $b

Incrementing/Decrementing Operators

Ex. ++ , --.

Logical Operators

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Ex. and, Or , xor , !, &&,||

String Operators

.(Concatenation) - Concatenate multiple strings.
.=(Concatenation and Assignment) - First concatenate First String and Second String, then assign the concatenated string to First String.

Array Operators

these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

Conditional Assignment Operators

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

?:(Ternary) - Returns the value of $x.
The value of $x is expr2 if expr1 = TRUE.
The value of $x is expr3 if expr1 = FALSE

Ex. $x = expr1 ? expr2 : expr3.

```
<?php
$user = "John";
  // if empty($user) = FALSE, set $status = "logged in"
  echo $status = (empty($user)) ? "not available" : "logged in";
?>
```

??(Null coalescing) - Returns the value of $x.
The value of $x is expr1 if expr1 exists, and is not NULL.
If expr1 does not exist, or is NULL, the value of $x is expr2.
Introduced in PHP 7

Ex. $x = expr1 ?? expr2.

```
<?php
  // variable $user is the value of $user
  // and 'anonymous' if it does not exist
  $user=NULL;
  echo $user = $user ?? "John";
  echo("<br>");
 $color='pink';
  // variable $color is "red" if $color does not exist or is null
  echo $color = $color ?? "red";
?>
```

2.5 Validating Form Data

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc.

These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form.

You need to validate a few things:

1. Empty String
2. Validate String
3. Validate Numbers
4. Validate Email
5. Validate URL
6. Input length

Empty String

Checks that the field is not empty. If the user leaves the required field empty, it will show an error message.

```
if (empty($_POST["name"])) {
   $errMsg = "Error! You didn't enter the Name.";
} else {
   $name = $_POST["name"];
}
```

Validate String

The field will contain only alphabets and whitespace.

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
   $ErrMsg = "Only alphabets and whitespace are allowed.";
} else {
   echo $name;
}
```

Validate Number

The field will only contain a numeric value.

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
   $ErrMsg = "Only numeric value is allowed.";
   echo $ErrMsg;
} else {
   echo $mobileno;
}
```

Validate Email

A valid email must contain @ and . symbols. PHP provides various methods to validate the email address.

```
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
   $ErrMsg = "Email is not valid.";
} else {
   echo "Your valid email address is: " .$email;
}
```

Input Length Validation

The input length validation restricts the user to provide the value between the specified range.

```
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);

if ( $length < 10 && $length > 10) {
   $ErrMsg = "Mobile must have 10 digits.";
        echo $ErrMsg;
} else {
   echo "Your Mobile number is: " .$mobileno;
}
```

Validate URL

The below code validates the URL of website provided by the user via HTML form.

```
$websiteURL = $_POST["website"];
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "URL is not valid";
  echo $websiteErr;
} else {
    echo "Website URL is: " .$websiteURL;
}
```

Button Click Validate

the user click on submit button and send the form data to the server one of the following method - get or post.

```
if (isset ($_POST['submit']) {
    echo "Submit button is clicked.";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        echo "Data is sent using POST method ";
    }
} else {
    echo "Data is not submitted";
}
```

2.6 What Are Arrays?

“An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.”

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

Indexed Array

index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1. $season=array("summer","winter","spring","autumn");

2. $season[0]="summer";
   $season[1]="winter";
   $season[2]="spring";
   $season[3]="autumn";

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:
Season are: summer, winter, spring and autumn

Associative Array

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1. $salary=array("Sono"=>"35000","John"=>"45000","Kartik"=>"20000");

2.
```
$salary["Ajay"]="350000";
$salary["Shital"]="450000";
$salary["Kartik"]="200000";
```

```
<?php
$salary["Ajay"]="350000";
$salary["Suniket"]="450000";
$salary["Kartik"]="200000";
echo " Ajay salary: ".$salary["Ajay "]."<br/>";
echo " Suniket salary: ".$salary["Suniket "]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
    ?>
```

Multidimensional Array

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.

Syntax

```
$emp = array
 (
 array(1,"sonoo",400000),
 array(2,"john",500000),
 array(3,"rahul",300000)
 );
```

Example

```
<?php
$emp = array
 (
 array(1,"sono",700000),
 array(2,"madhav",600000),
 array(3,"rahul",400000)
 );

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

```
}
?>
```

Output:

```
1 sono 700000
2 madhav 600000
3 rahul 400000
```

2.7 For and While Loops

PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.

It allows users to put all the loop related statements in one place.

Syntax

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

initialization - Initialize the loop counter value. The initial value of the for loop is done only once. This parameter is optional.

condition - Evaluate each iteration value. The loop continuously executes until the condition is false. If TRUE, the loop execution continues, otherwise the execution of the loop ends.

Increment/decrement - It increments or decrements the value of the variable.

Example

```
<?php
for($n=1;$n<=5;$n++){
echo "$n<br/>";
}
?>
```

Output:

```
1
2
3
4
5
```

While Loop
PHP while loop can be used to traverse set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of loop.

It should be used if the number of iterations is not known.

The while loop is also called an Entry control loop because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.

Syntax
```
while(condition){
//code to be executed
}
```

OR

Example
```
<?php
$n=1;
while($n<=5){
echo "$n<br/>";
$n++;
}
?>
```

OR

```
<?php
$n=1;
while($n<=10):
echo "$n<br/>";
$n++;
endwhile;
?>
```

The End