PHP Operators

Session 8

◆ *Explain the use of arithmetic operators*

◆ *Explain the use of logical operators*

◆ *Explain the use of relational operators*

◆ *Explain the use of bitwise operators*

◆ *Explain the use of assignment operators*

◆ *Explain the use of string operators*

◆ *Explain the use of increment and decrement operators*

◆ *Explain the use of conditional or ternary operators*

◆ *Explain the precedence of operators*

◆ All programming languages use operators

◆ Operators

  ◈ Are pre-defined symbols that perform specific actions on objects called operands

  ◈ Are assigned a precedence value indicating the order in which the operators are evaluated

◆ Operators are classified as follows:

◈ **Unary Operator** - acts on only one operand in an expression

◈ **Binary Operator** - has two operands and performs different arithmetic and logical operations

◈ **Conditional or Ternary Operator** – has three operands and evaluates the second or third expression depending on the result of the first expression

◆ Are binary operators that work only on numeric operands

◆ If operands are non-numeric values such as strings, Booleans, nulls, or resources, they are converted to their numeric equivalents

Table lists the arithmetic operators supported by PHP

Operator	Name	Description
+	Addition	Returns the sum of the operands
−	Subtraction	Returns the difference between the two operands
*	Multiplication	Returns the product of two operands
/	Division	Returns the quotient after dividing the first operand by the second operand
%	Modulus	Returns the remainder after dividing the first operand by the second operand

◆ **Displaying a mathematical expression**

❖ Enter the code as shown in a PHP script named `math-exp.php`

Snippet

```php
<?php
$var1 = 21-4*4;
echo "The result of 21-4*4 is $var1.";
?>
```

Displays the following output:
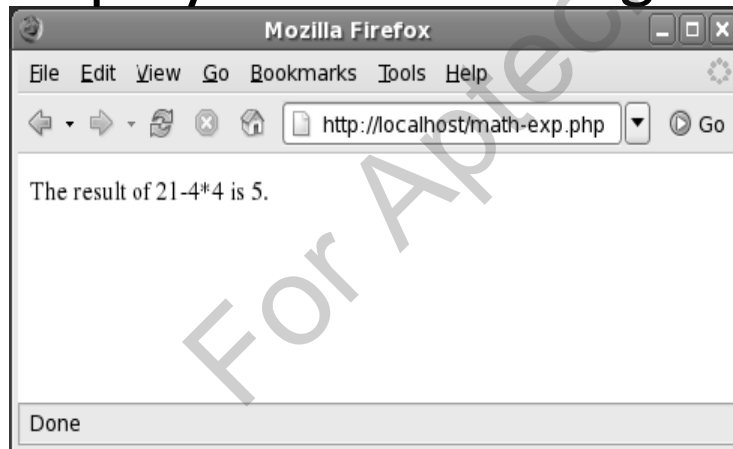
In the code, if you follow the Brackets Order Division Multiply Add Subtract (BODMAS) rule of mathematics, the result will be 8.

PHP follows the BODMAS rule for calculation.

- ## Displaying the use of arithmetic operators

  - Enter the code in a PHP script named **sumarithmatic.php**

Snippet

```php
<?php
$VAR1=8;
$VAR2=10;
//Addition
$SUM=$VAR1+$VAR2;
//Subtraction
$DIFFERENCE =$VAR1-$VAR2;
//Multiplication
$PRODUCT = $VAR1*$VAR2;
//Division
$QUOTIENT =$VAR1/$VAR2;
//Modulus
$REMAINDER = $VAR1%$VAR2;
```

```
echo "Addition of 5 and 10 is ".$SUM."<br>";
echo "Subtraction of 10 from 5 is ".$DIFFERENCE."<br>";
echo "Multiplication of 5 and 10 is ".$PRODUCT."<br>";
echo "Division of 5 and 10 is ".$QUOTIENT."<br>";
echo "Modulus of 5 and 10 is ".$REMAINDER."<br>";
?>
```

Displays the following output:
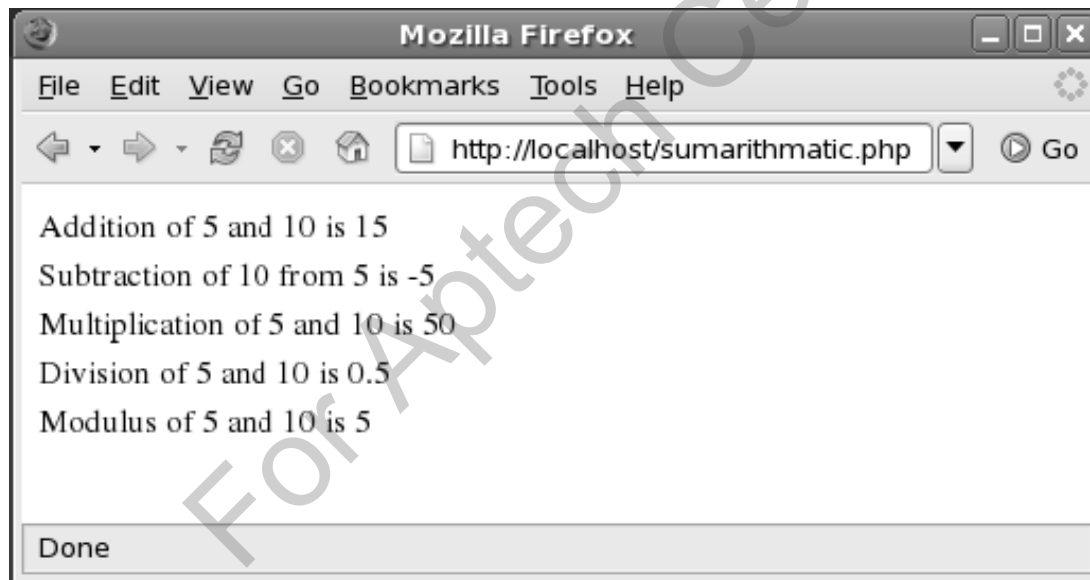
◆ Compares two operands and returns either a `true` or a `false` value

◆ The operands can either be numbers or string values

Table lists relational operators along with its description

Operator	Name	Description
==	Equal to	Returns true if both the operands are equal
===	Identical	Returns true if both the operands are equal and are of the same data type (Introduced in PHP 4)
!=	Not equal to	Returns true if the first operand is not equal to the second operand
<>	Not equal to	Returns true if the first operand is not equal to the second operand
!==	Not Identical	Returns true if the first operand is not equal to the second operand or they are not of the same data type (Introduced in PHP 4)
<	Less than	Returns true if the first operand is less than the second operand
<=	Less than or equal to	Returns true if the first operand is less than or equal to the second operand
>	Greater than	Returns true if the first operand is greater than the second operand
>=	Greater than or equal to	Returns true if the first operand is greater than or equal to the second operand

- Showing the difference between '== 'equal and '===' identical relational operators in PHP

Snippet

```php
<?php
if("10" == 10)
    echo "YES";
    else
    echo "NO";
    ?>
```

Displays the following output:
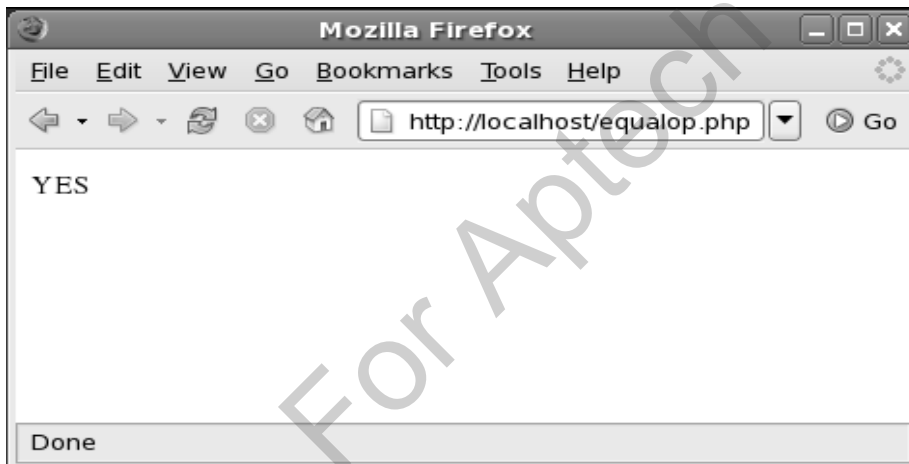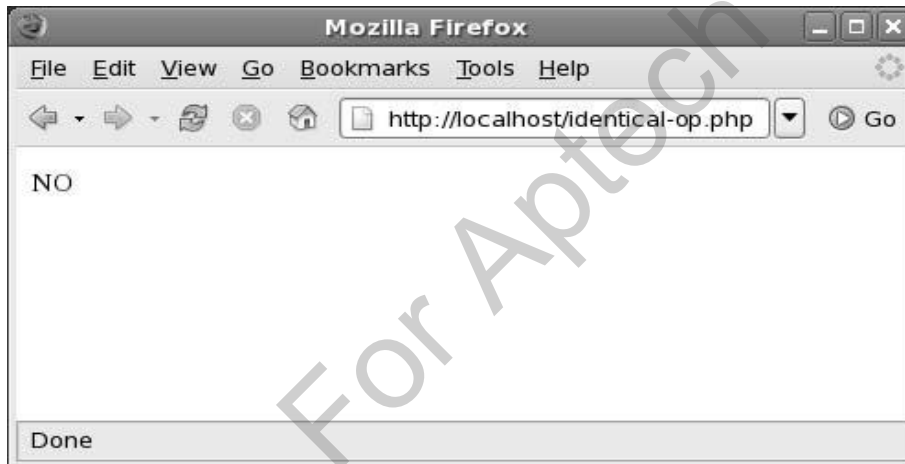

The code will print YES because the values of the operands are equal.

◆ Showing the difference between '== 'equal and '===' identical relational operators in PHP

Snippet

```php
<?php
if("10" === 10)
    echo "YES";
    else
    echo "NO";
    ?>
```
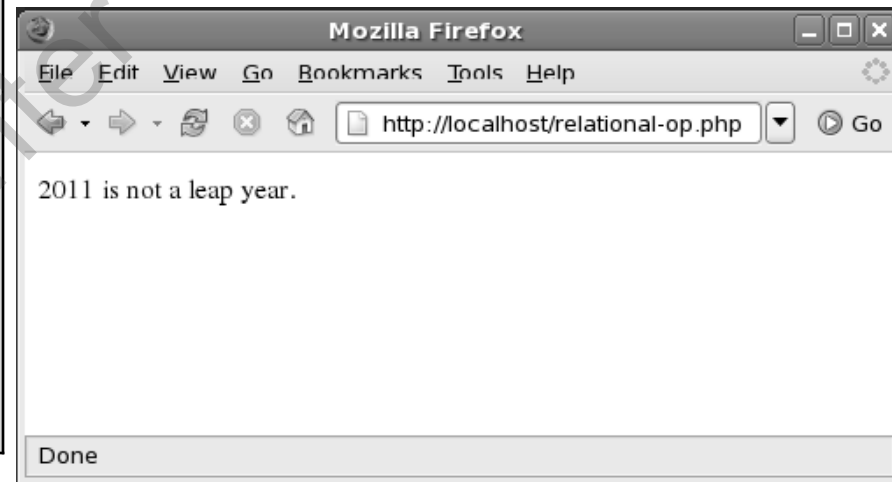
Displays the following output:

The code will print NO because although values of both operands are same, their data types are different. "10" is a string while 10 is an integer.

◆ **Checking whether the given year is a leap year or not:**

◈ Enter the code as shown in a script named **relational-op.php**

Snippet

```php
<?php
$y = 2011;
if(($y%4==0 && $y%100!=0) || ($y%400 == 0))
{
    echo "$y is a leap year. <br />";
}
else
{
echo "$y is not a leap year. <br />";
}
?>
```

Displays the following output:

Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/relational-op.php

2011 is not a leap year.

Done

The code uses an `if` conditional statement to execute a block of code only when the specified condition is `true` else it executes the block of code in the body of the `else` statement.

The output of the code will be 2011 is not a leap year.

- ◆ Enable to combine two or more expressions in a condition
- ◆ Evaluates the expression and returns a Boolean value of either `true` **or** `false`

Table lists various logical operators

Operator	Name	General Form	Description
AND &&	Logical AND operator	Expression1 AND Expression2 Expression1 && Expression2	Returns true only if both the expressions are true
OR ||	Logical OR operator	Expression1 OR Expression2 Expression1 || Expression2	Returns true if any one of the expression is true
XOR	Logical XOR operator	Expression1 XOR Expression2	Returns true if either Expression 1 or Expression 2 is true, but not both
!	Logical NOT operator	!Expression	Returns true only if the condition is not true

◆ Using a logical `AND` operator to check if a student's percentage is greater than 60 and the year of passing is 2003

- ◈ Enter the code as shown in a PHP script named **logical_and.php**

Snippet

```php
<?php
$Percentage = 70;
$Year = "2003";
if ($Percentage>60 AND $Year=="2003")
{
    echo "Eligible for admission.";
    }
    ?>
```

Displays the following output:
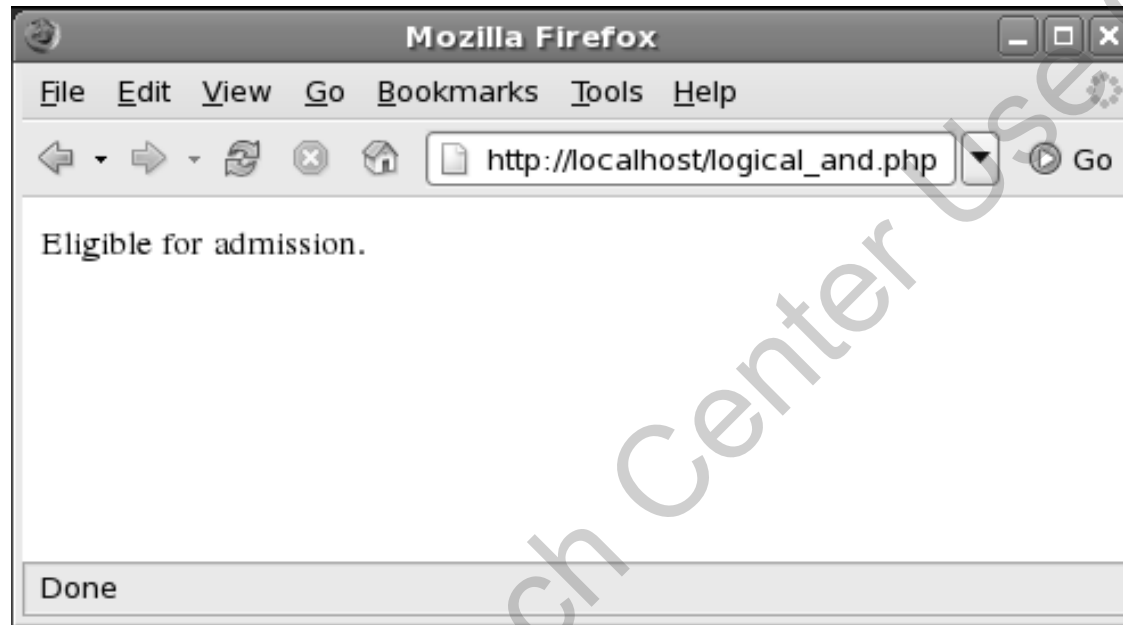
In the code, the `AND` operator requires both the expressions to be `true`.

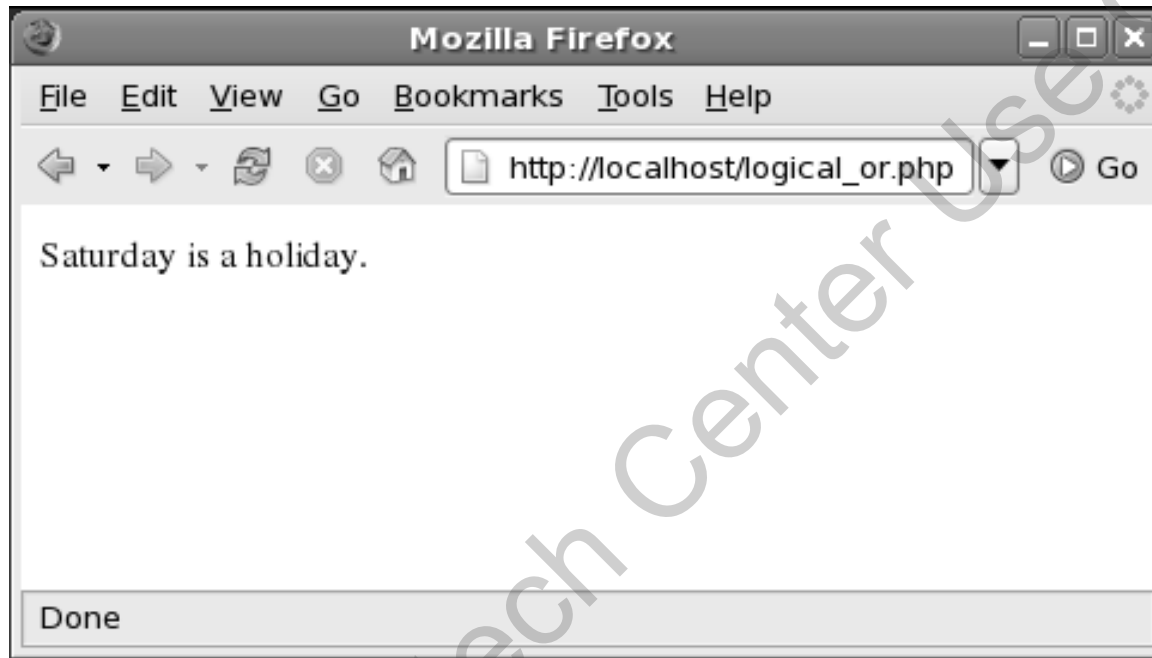Only then, the block of code following the `if` statement is executed.

◆ **Displaying the use of the OR operator**

⬦ Enter the code as shown in a PHP script named, **logical_or.php**

```php
<?php
$day1="Saturday";
if(($day1=="Saturday")||($day1=="Sunday"))
{
    echo "$day1 is a holiday.";
    }
    else
    {
    echo "$day1 is a working day.";
    }
    ?>
```

Displays the following output:

In the code, the `OR` operator requires any one of the expressions to be `true`, and executes the block of code after the `if` statement.

- Operate on the bitwise representation of their operands

- Work on small-scale binary representation of data

- If the operand is a string, the operation is performed only after converting it to its corresponding integer representation

Table lists bit-wise operators

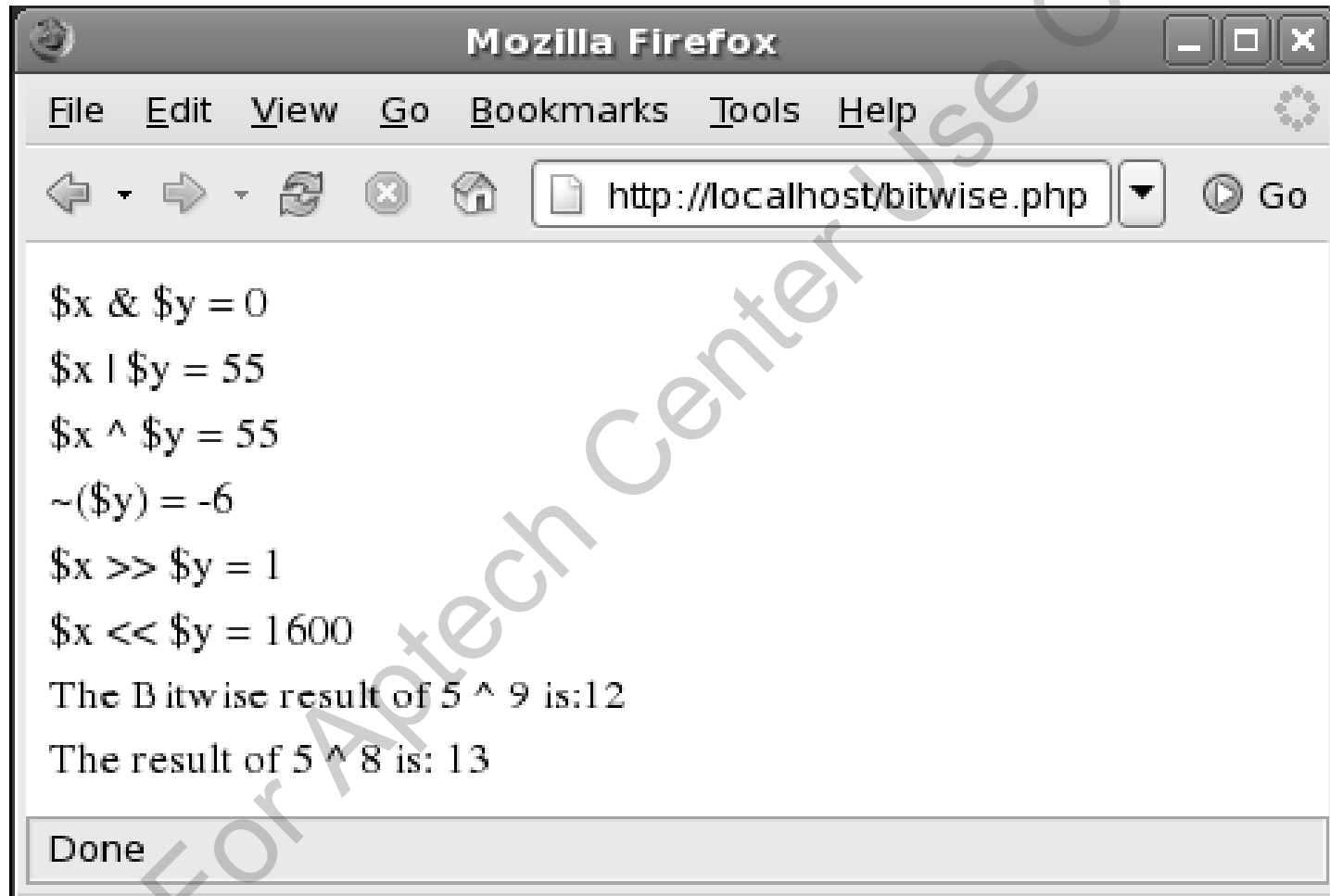| Operator	Name	General Form	Description
&	AND	Operand1 & Operand2	Compares two bits and sets the result to 1 if both the bits are 1 and 0 otherwise
|	OR	Operand1 | Operand2	Compares two bits and sets the result to 1 if either of the bits are 1 and 0 otherwise
^	EXCLUSIVE-OR	Operand1 ^ Operand2	Compares two bits and sets the bit to 1 if the bits are different and 0 otherwise
~	COMPLEMENT	~ Operand	Compares and sets the 0 bits to 1 and vice versa
<<	SHIFT LEFT	Operand1 << Operand2	Shifts the bits of Operand1, Operand2 steps to the left (each step means "multiply by two")
>>	SHIFT RIGHT	Operand1 >> Operand2	Shifts the bits of Operand1, Operand2 steps to the right (each step means " divide by two")

Table lists binary comparisons of bit-wise operators

X Y	X & Y	X | Y	~ X	~ Y	X ^ Y
1 1	1	1	0	0	0
1 0	0	1	0	1	1
0 1	0	1	1	0	1
0 0	0	0	1	1	0

- Displaying the use of bitwise operations

  - Enter the code in a PHP script named **bitwise.php**

Snippet

```php
<?php
$x= 50;
$y= 5;
echo "\$x & \$y = ".($x & $y)."<br>";
echo "\$x | \$y = ".($x | $y)."<br>";
echo "\$x ^ \$y = ".($x ^ $y)."<br>";
echo "~(\$y) = ".~$y."<br>";
//x is divided by 2 y times
echo "\$x >> \$v = ".($x >> $y)."<br>";
//x is multiplied by 2 y times
echo "\$x << \$y = ".($x << $y)."<br>";
//Converts  the  operands  to  their  ASCII  values  first
('5'(ascii 53))^ ('9' (ascii 57))
echo "The Bitwise result of 5 ^ 9 is:".(5 ^ 9)."<br>";
//Converts "8" to perform the operation (5 ^ ((int)"8")
echo "The result of 5 ^ 8 is: ".(5 ^ 8). "<br>";
?>
```

Displays the following output:

- Enables to assign a value to a variable
- The '=' sign is the assignment operator

Syntax

```
expression 1 = expression 2;
```

- Operand on the left side of the assignment operator '=' should be a variable

- The assignment operation can be performed by:
    - **Value** - copies the value of expression 2 and assigns it to expression 1
    - **Reference** - assigns a reference of expression 2 to expression 1

- The basic assignment operator can also be used as a shorthand operators (combined operator) in conjunction with arithmetic and string operators

- The shorthand operators are +=, -=, *=, /=, %=, and .=

Table displays various examples with shorthand operators

combihand	Expression	Description
$a+= $b	$a = $a + $b	Adds $a and $b and assigns the result to $a
$a-= $b	$a = $a -$b	Subtracts $b from $a and assigns the result to $a
$a*= $b	$a = $a*$b	Multiplies $a and $b and assigns the result to $a
$a/= $b	$a = $a/$b	Divides $a by $b and assigns the quotient to $a
$a%= $b	$a = $a%$b	Divides $a by $b and assigns the remainder to $a
$a.= $b	$a=$a.$b	Concatenates $b with $a and assigns the result to $a

◆ **Using shorthand operators**
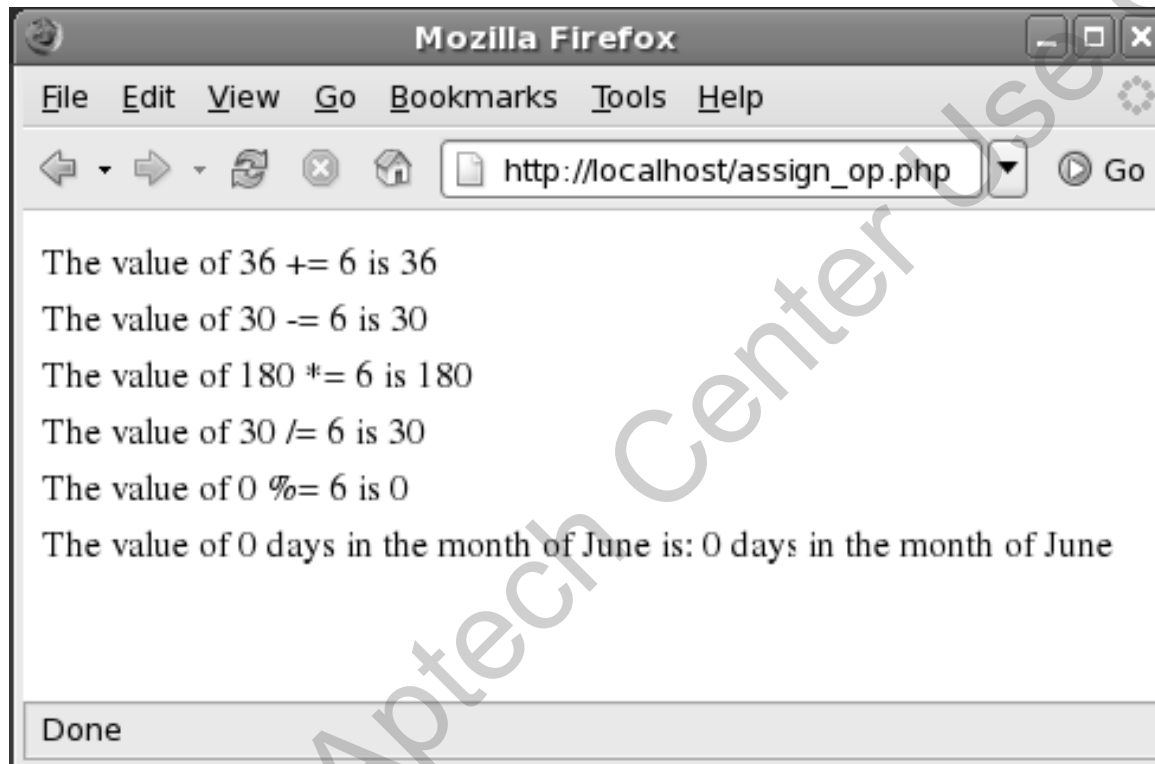
  ◈ Enter the code in a PHP script named **assign_op.php**

Snippet

```php
<?php
$a = 30;
$b = 6;
$a+= $b;
echo "The value of $a += $b is ". $a ."<br>";
$a-= $b;
echo "The value of $a -= $b is ". $a ."<br>";
$a*= $b;
echo "The value of $a *= $b is ". $a ."<br>";
$a/= $b;
echo "The value of $a /= $b is ". $a ."<br>";
$a%= $b;
echo "The value of $a %= $b is ". $a ."<br>";
$a.= " days in the month of June";
echo "The value of $a is: " . $a ."<br>";
?>
```

Displays the following output:

```
Mozilla Firefox

File  Edit  View  Go  Bookmarks  Tools  Help

http://localhost/assign_op.php          Go

The value of 36 += 6 is 36

The value of 30 -= 6 is 30

The value of 180 *= 6 is 180

The value of 30 /= 6 is 30

The value of 0 %= 6 is 0

The value of 0 days in the month of June is: 0 days in the month of June

Done
```

- Operate only on variables

- Cause a variable to change its value

- The increment operators increase the value of the operand by one

- The decrement operators decrease the value of the operand by one

Table lists increment and decrement operators

Operand	Operator Name	Description
++$a	Pre-increment	Increments the operand value by one and then returns this new value to the variable
$a++	Post-increment	Returns the value to the variable and then increments the operand by one
--$a	Pre- decrement	Decrements the operand by one and then returns this new value to the variable
$a--	Post decrement	Returns the value to the variable and then decrements the operand by one

◆ Displaying the use of the increment and decrement operators on a variable $A

  ◈ Enter the code as shown in a PHP script named **inc_dec.php**

Snippet

```php
<?php
$A=15;
echo "Pre increment value of A is ".++$A."<br/>";
echo "Post increment value of A is ".$A++."<br/>";
echo "Value of A post increment is ".$A."<br/>";
echo "Pre decrement value of A is ".--$A."<br/>";
echo "Post decrement value of A is ".$A--."<br/>";
echo "Value of A post decrement is ".$A--."<br/>";
?>
```
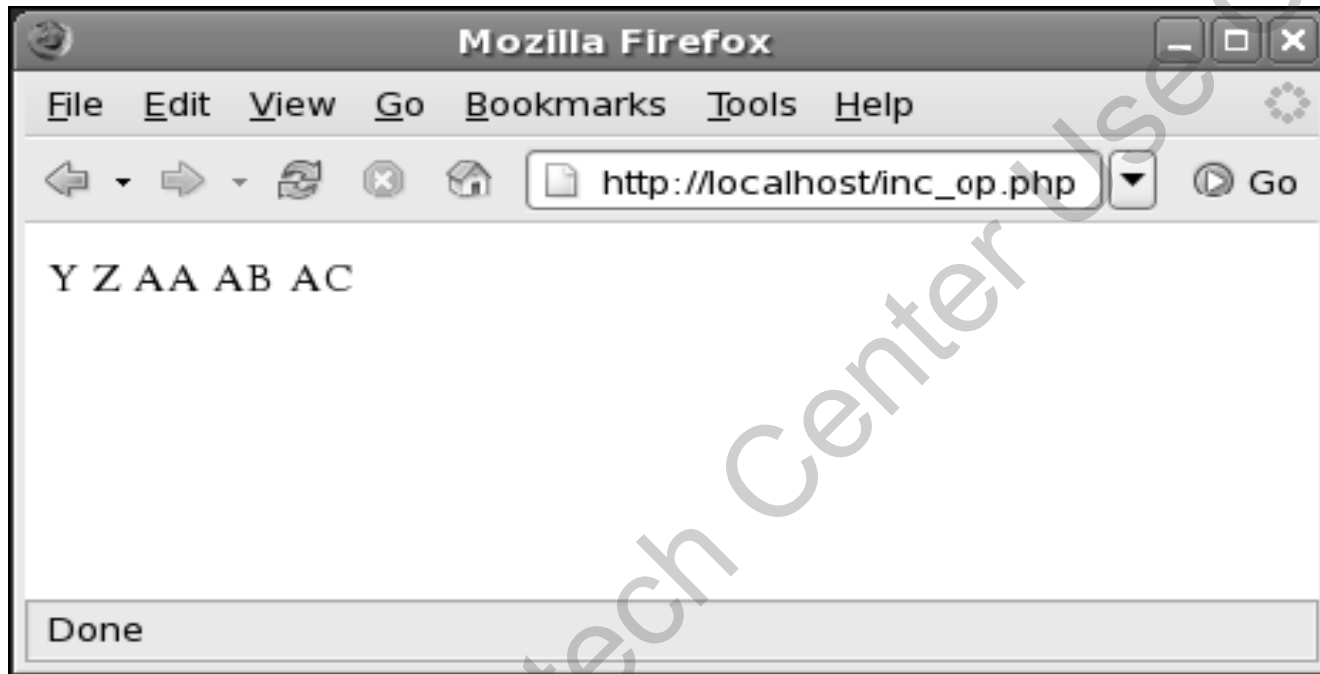
Displays the following output:
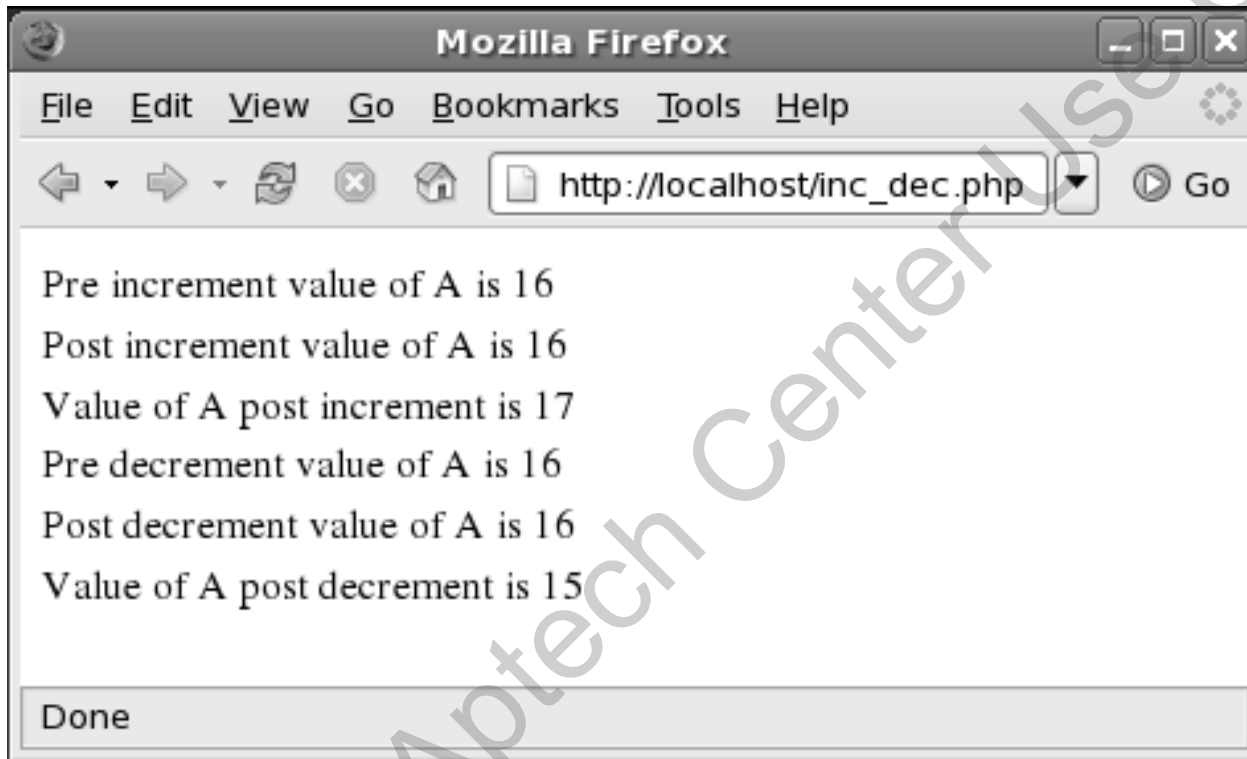
◆ Displaying the use of increment operator on a character variable

  ◈ Enter the code as shown in a PHP script named **inc_op.php**

Snippet

```php
<?php
$i = 'X';
for ($n=0; $n<5; $n++)
{
echo ++$i . "\n";
}
?>
```

◆ Displays the following output:

In the code, the value of the variable `$i` is incremented five times using a for loop statement and each value is printed. Here `++$i`, i.e., 'X'+1 returns 'Y' followed by 'Z' and 'Z'+1 returns 'AA' and 'AA'+1 returns 'AB'.

◆ Operate only on character data

◆ Non-string operand is first converted before the operation is executed

Table lists string operators

Operator	Name	Description
.	Concatenation	Returns a concatenated string
. =	Concatenating assignment	Appends the argument on the right side of the operator to the arguments on the left side

◆ Displaying the concatenation of the strings WELCOME and FRIENDS using the Concatenation operator

   ◈ Enter the code as shown in a PHP script named **concat.php**

Snippet

```php
<?php
$A="WELCOME ";
$B="FRIENDS!";
$C=$A.$B;
echo "The concatenated string is $C";
?>
```

◆ Displays the following output:

In the code, the values of the variables $A and $B are concatenated and stored in a variable $C.

It returns the string WELCOME FRIENDS! as the output.

◆ Assigning a value to a variable using the concatenating assignment operator

    ◈ Enter the code as shown in a PHP script named **assign_concat.php**

Snippet

```php
<?php
$A="WELCOME";
$A.= " FRIENDS!";
echo " The concatenated string is $A";
?>
```

◆ Displays the following output:
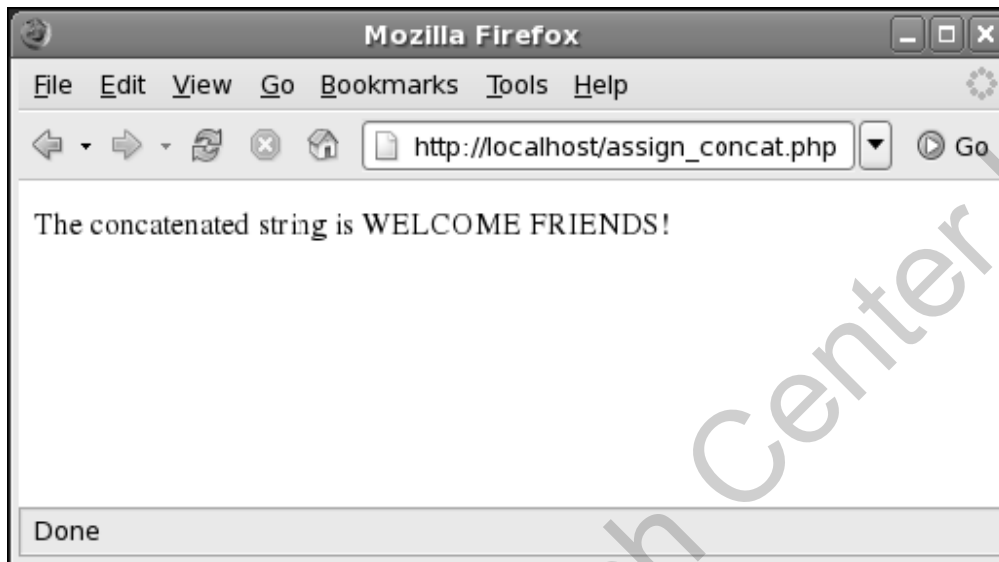
In the code, the argument FRIENDS! is appended on the right side of $A containing the argument WELCOME.

The string returned is WELCOME FRIENDS!.

◆ An alternative to the `if-else` statement

◆ Evaluates an expression for a `true` or `false` value and then executes one of the two statements depending upon the result of evaluation

Syntax

```
$var1 = ($var2 = value1) ? expr1 : expr2
```

Where,

◈ The operator evaluates the `$var1` and if it is `true`, `expr 1` is evaluated and if it is `false`, `expr 2` is evaluated

◆ ## Displaying the use of a conditional operator

◈ Enter the code as shown in a PHP script named **condition.php**

Snippet

```php
<?php
$age = 15;
$category = ($age < 16) ? 'Child' : 'Adult';
echo "The result of the conditional operator is: ";
echo $category;
?>
```

◆ Displays the following output:

The code evaluates the condition ($age < 16).

If it is `true`, then the value Child is assigned to `$category`, else the value Adult is assigned to `$category`.

◆ The order of precedence to evaluate the expression in PHP is:

  ◈ Operators with higher precedence are evaluated first in an expression

  ◈ Operator precedence can be overridden by using parenthesis. The expression within the parenthesis is evaluated first

  ◈ If two operators of the same precedence are encountered, the expression will be evaluated from left to right

Table lists the precedence of operators in PHP, with the highest precedence operators listed at the top

Precedence	Category	Operators	Associativity
	Unary	!, ++ ,--	Right to left
Highest Precedence	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
	Logical AND	AND	Left to right
	Logical XOR	XOR	Left to right
	Logical OR	OR	Left to right
Lowest Precedence	Comma	,	Left to right

- Operator is any symbol that performs an operation on an operand

- An operator enables you to work on variables, strings, and numbers and control the program flow

- The types of operators supported in PHP are arithmetic, logical, relational, bitwise, assignment, string, conditional, and increment and decrement operators

- The arithmetic operators work with numbers and are used to execute mathematical operations. PHP follows the BODMAS rule for operator precedence

- The relational operators compare two operands and determine the relationship between operands

- The logical operators evaluate multiple conditions and combine two or more test expression in a condition, returning a Boolean value

- The Bitwise operators enable comparison and manipulation of operands and operate on the bits of an operand

- The assignment operator enables assignment of values to a variable and defines the operand on the left-hand side to the value on the right-hand side

- The increment and decrement operators enable to increase or decrease value of an operand by one

◆ The conditional or ternary operator is an alternative to the if-else statement. It evaluates a condition and executes one of the two statements depending on the true or false result of the condition

◆ The concatenation operator combines two or more strings into a single string

◆ In a complex expression, operator precedence indicates the order in which the operands must be evaluated. PHP evaluates operators with high precedence before operators with low precedence