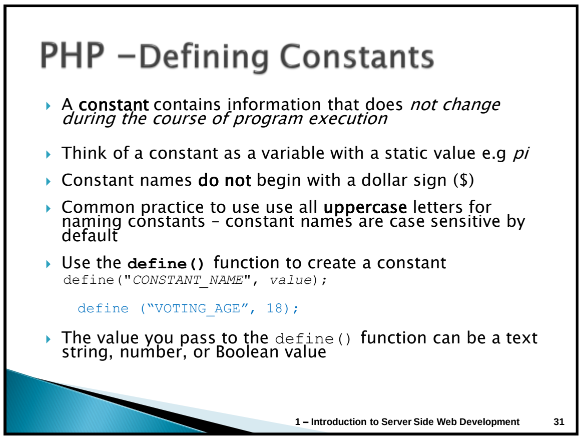
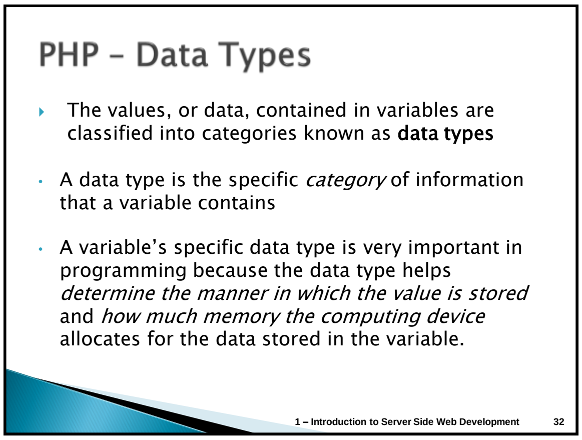


PHP –Defining Constants

- A **constant** contains information that does *not change during the course of program execution*
- Think of a constant as a variable with a static value e.g *pi*
- Constant names **do not** begin with a dollar sign ($)
- Common practice to use use all **uppercase** letters for naming constants – constant names are case sensitive by default
- Use the **`define()`** function to create a constant
  `define("CONSTANT_NAME", value);`

  ```
  define ("VOTING_AGE", 18);
  ```
- The value you pass to the `define()` function can be a text string, number, or Boolean value

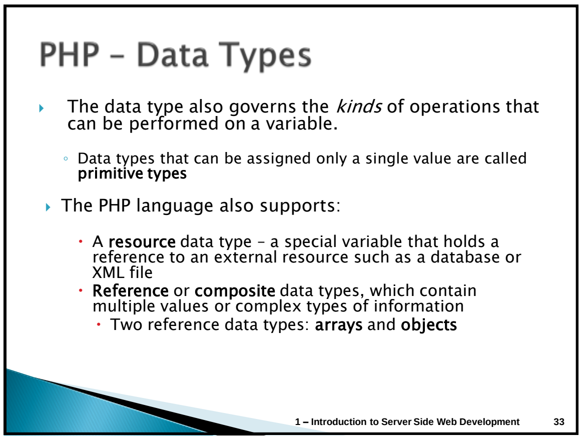
1 – Introduction to Server Side Web Development 31



PHP – Data Types

- The values, or data, contained in variables are classified into categories known as **data types**
- A data type is the specific *category* of information that a variable contains
- A variable's specific data type is very important in programming because the data type helps *determine the manner in which the value is stored* and *how much memory the computing device* allocates for the data stored in the variable.

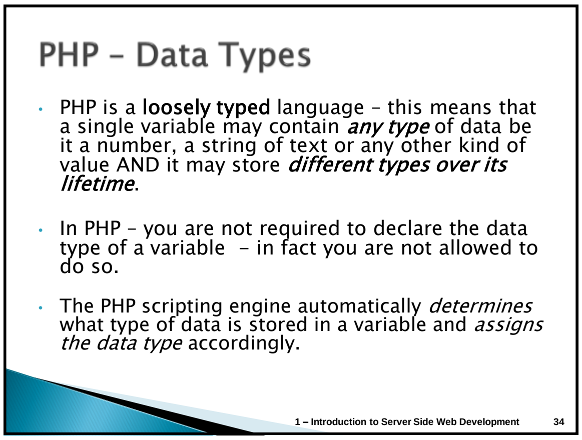
1 – Introduction to Server Side Web Development 32



PHP – Data Types

- The data type also governs the *kinds* of operations that can be performed on a variable.
  - Data types that can be assigned only a single value are called **primitive types**
- The PHP language also supports:
  - A **resource** data type – a special variable that holds a reference to an external resource such as a database or XML file
  - **Reference** or **composite** data types, which contain multiple values or complex types of information
    - Two reference data types: **arrays** and **objects**

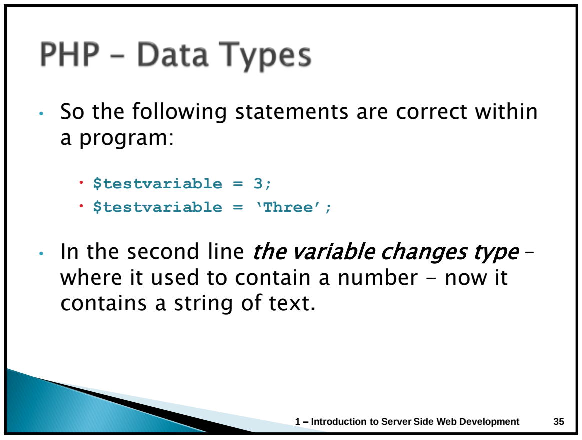
1 – Introduction to Server Side Web Development 33



PHP – Data Types

- PHP is a **loosely typed** language – this means that a single variable may contain ***any type*** of data be it a number, a string of text or any other kind of value AND it may store ***different types over its lifetime***.
- In PHP – you are not required to declare the data type of a variable – in fact you are not allowed to do so.
- The PHP scripting engine automatically *determines* what type of data is stored in a variable and *assigns the data type* accordingly.

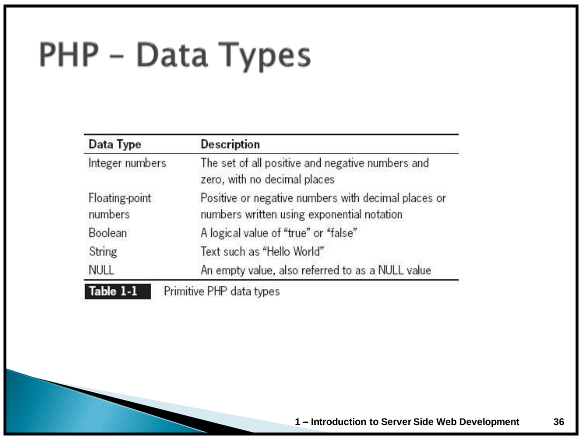
1 – Introduction to Server Side Web Development 34



PHP – Data Types

- So the following statements are correct within a program:
  - `$testvariable = 3;`
  - `$testvariable = 'Three';`
- In the second line ***the variable changes type*** – where it used to contain a number – now it contains a string of text.

1 – Introduction to Server Side Web Development 35

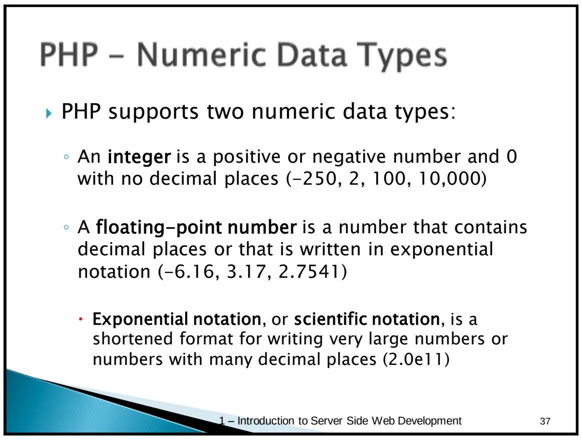


PHP – Data Types

Data Type	Description
Integer numbers	The set of all positive and negative numbers and zero, with no decimal places
Floating-point numbers	Positive or negative numbers with decimal places or numbers written using exponential notation
Boolean	A logical value of "true" or "false"
String	Text such as "Hello World"
NULL	An empty value, also referred to as a NULL value

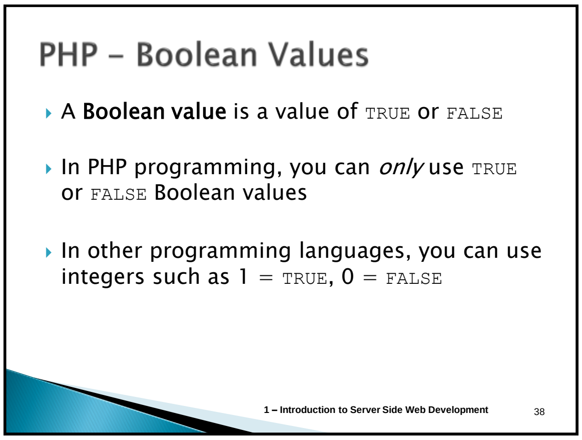
Table 1-1 Primitive PHP data types

1 – Introduction to Server Side Web Development 36



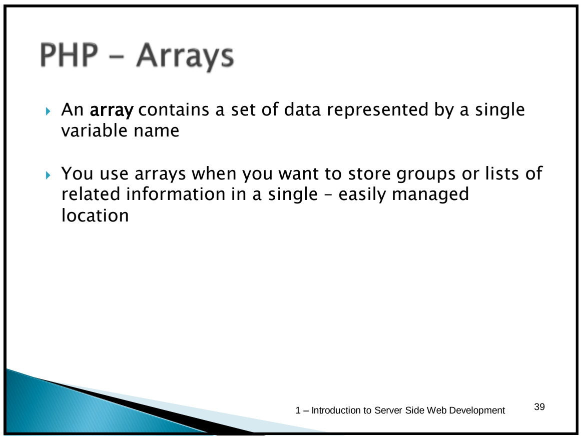
PHP – Numeric Data Types

- PHP supports two numeric data types:
  - An **integer** is a positive or negative number and 0 with no decimal places (-250, 2, 100, 10,000)
  - A **floating-point number** is a number that contains decimal places or that is written in exponential notation (-6.16, 3.17, 2.7541)
    - **Exponential notation**, or **scientific notation**, is a shortened format for writing very large numbers or numbers with many decimal places (2.0e11)



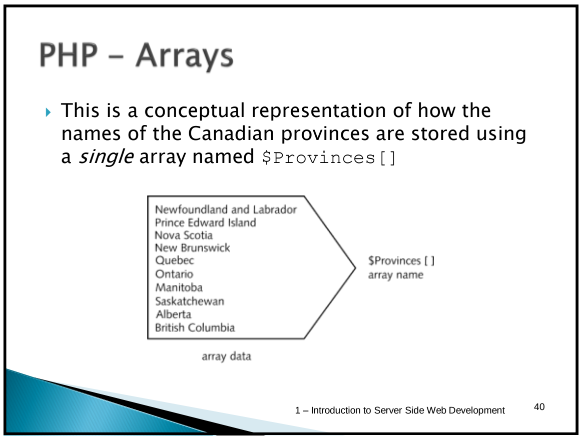
PHP – Boolean Values

- A **Boolean value** is a value of `TRUE` or `FALSE`
- In PHP programming, you can *only* use `TRUE` or `FALSE` Boolean values
- In other programming languages, you can use integers such as 1 = `TRUE`, 0 = `FALSE`



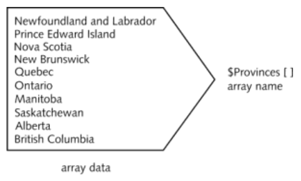
PHP – Arrays

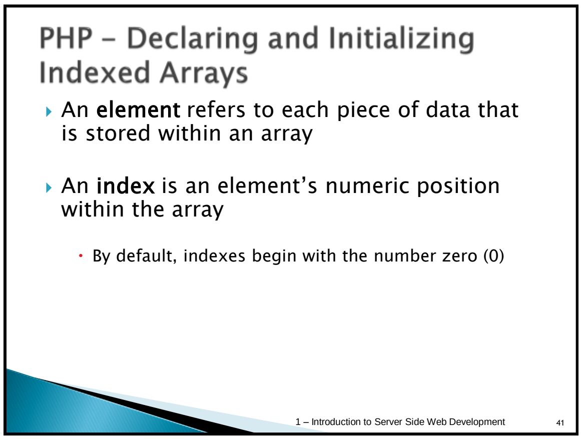
- An **array** contains a set of data represented by a single variable name
- You use arrays when you want to store groups or lists of related information in a single – easily managed location



PHP – Arrays

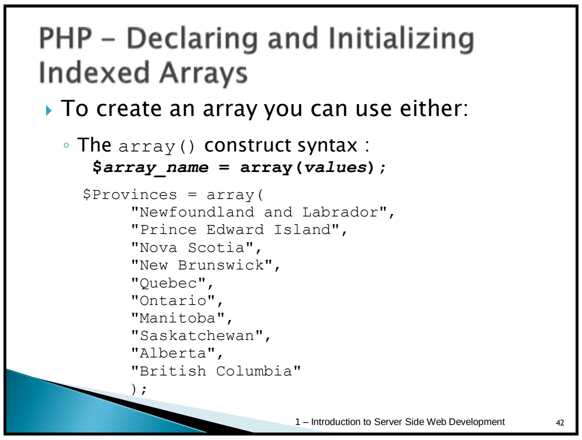
- This is a conceptual representation of how the names of the Canadian provinces are stored using a *single* array named `$Provinces[]`





PHP – Declaring and Initializing Indexed Arrays

- An **element** refers to each piece of data that is stored within an array
- An **index** is an element's numeric position within the array
    - By default, indexes begin with the number zero (0)



PHP – Declaring and Initializing Indexed Arrays

- To create an array you can use either:
  - The `array()` construct syntax :

```
  $array_name = array(values);
$Provinces = array(
      "Newfoundland and Labrador",
      "Prince Edward Island",
      "Nova Scotia",
      "New Brunswick",
      "Quebec",
      "Ontario",
      "Manitoba",
      "Saskatchewan",
      "Alberta",
      "British Columbia"
      );
```

PHP – Declaring and Initializing Indexed Arrays

- Or
  - Array name and brackets syntax:***$array_name*[ ]**

```
$Provinces[] = "Newfoundland and Labrador";
$Provinces[] = "Prince Edward Island";
$Provinces[] = "Nova Scotia";
$Provinces[] = "New Brunswick";
$Provinces[] = "Quebec";
$Provinces[] = "Ontario";
$Provinces[] = "Manitoba";
$Provinces[] = "Saskatchewan";
$Provinces[] = "Alberta";
$Provinces[] = "British Columbia";
```

PHP – Accessing Array Elements

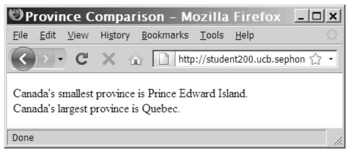
- An element is referenced by enclosing its **index** in brackets at the end of the array name:

```
$Provinces[1]
```

- refers to the `Prince Edward Island` state

PHP – Accessing Element Information

```
echo "<p>Canada's smallest province is
  $Provinces[1].<br />";
echo "Canada's largest province is
  $Provinces[4].</p>";
```


Province Comparison – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephon
Done

Canada's smallest province is Prince Edward Island.
Canada's largest province is Quebec.

Output of elements in the `$Provinces[]` array

PHP – Arrays

- In PHP – *each array space* can contain **any type of value**

```
$myarray = array('one', 2, '3');
```

- Contains three values:

⇨	'one'	a string
⇨	2	a number
⇨	'3'	a string

PHP – Modifying Arrays

```
$myarray = array('one', 2, '3');
```

- You can use an array index in square brackets to :
  - add new elements
  - assign new values to existing elements

```
$myarray[1] = 'two';    // Assign a new value
$myarray[3] = 'four';   // Create a new element
```

PHP – Accessing Element Information in Associative Arrays

- To access an array element in an associative array – use the appropriate string index in square brackets after the array variable name:

```
echo 'My birthday is: ' . $birthdays['Kevin'];
```

concatenation operator

associated string index

array variable name

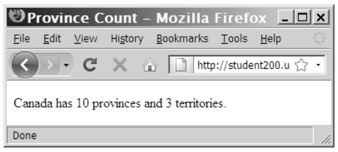
PHP – Accessing Element Information in Arrays

- Use the **`count()`** function to find the total number of elements in an array

```
$Provinces = array("Newfoundland and Labrador",
"Prince Edward
Island", "Nova Scotia", "New Brunswick", "Quebec",
"Ontario", " Manitoba", "Saskatchewan", "Alberta",
"British Columbia");

  $Territories = array("Nunavut", "Northwest
  Territories", "Yukon  Territory");

  echo "<p>Canada has ", count($Provinces), "
  provinces and ",
         count($Territories), " territories.</p>";
```

PHP – Accessing Element Information in Arrays

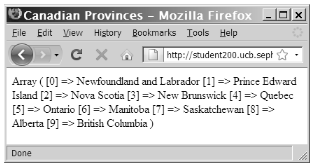
Output of the `count()` function


Province Count – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.u
Done

```
Canada has 10 provinces and 3 territories.
```

PHP – Accessing Element Information in Arrays

Output of the `$Provinces[ ]` array with the **`print_r()`** function:


Canadian Provinces – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sept
Done

```
Array ( [0] => Newfoundland and Labrador [1] => Prince Edward
Island [2] => Nova Scotia [3] => New Brunswick [4] => Quebec
[5] => Ontario [6] => Manitoba [7] => Saskatchewan [8] =>
Alberta [9] => British Columbia )
```

PHP – Modifying Elements

- To modify an array element. include the index for an individual element of the array:

```
$HospitalDepts = array(
    "Anesthesia",       // first element(0)
    "Molecular Biology",// second element (1)
    "Neurology");       // third element (2)
```

To change the first array element in the `$HospitalDepts[]` array from “Anesthesia” to “Anesthesiology” use:

```
$HospitalDepts[0] = "Anesthesiology";
```

PHP – Avoiding Assignment Notation Pitfalls

- **Assigns** the string “Hello” to a variable named `$list`

```
$list = "Hello";
```

- **Assigns** the string “Hello” to a new element appended to the end of the `$list` array

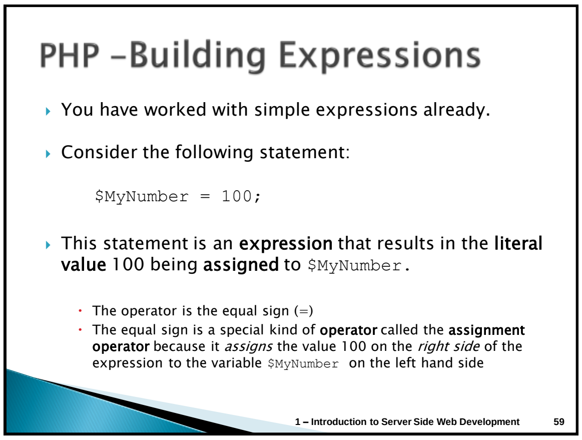
```
$list[] = "Hello";
```

- **Replaces** the value stored in the first element (index 0) of the $list array with the string “Hello”

```
$list[0] = "Hello";
```

PHP –Building Expressions

- An **expression** is a literal value or variable that can be evaluated by the PHP scripting engine to produce a result
- **Operands** are variables and literals contained in an expression
- A **literal** is a static value such as a literal string or a number
- **Operators** are symbols (+) (*) that are used in expressions to manipulate operands



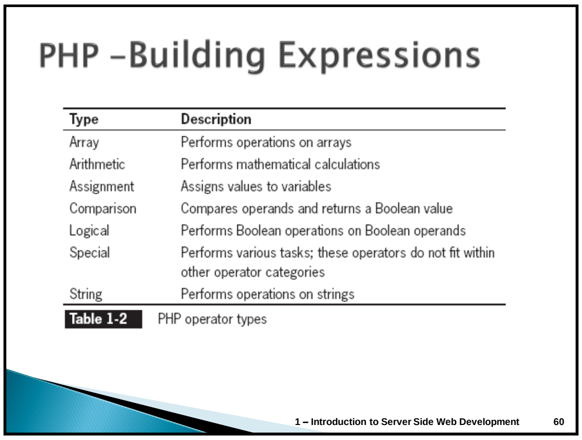
PHP –Building Expressions

- You have worked with simple expressions already.
- Consider the following statement:

```
$MyNumber = 100;
```

- This statement is an **expression** that results in the **literal value** 100 being **assigned** to `$MyNumber`.
  - The operator is the equal sign (=)
  - The equal sign is a special kind of **operator** called the **assignment operator** because it *assigns* the value 100 on the *right side* of the expression to the variable `$MyNumber` on the left hand side

1 – Introduction to Server Side Web Development 59

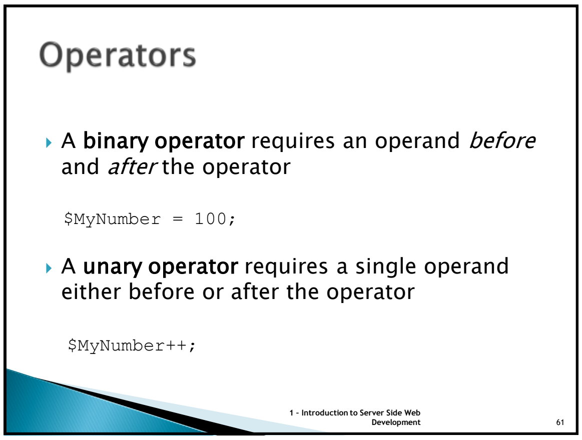


PHP –Building Expressions

Type	Description
Array	Performs operations on arrays
Arithmetic	Performs mathematical calculations
Assignment	Assigns values to variables
Comparison	Compares operands and returns a Boolean value
Logical	Performs Boolean operations on Boolean operands
Special	Performs various tasks; these operators do not fit within other operator categories
String	Performs operations on strings

Table 1-2 PHP operator types

1 – Introduction to Server Side Web Development 60



Operators

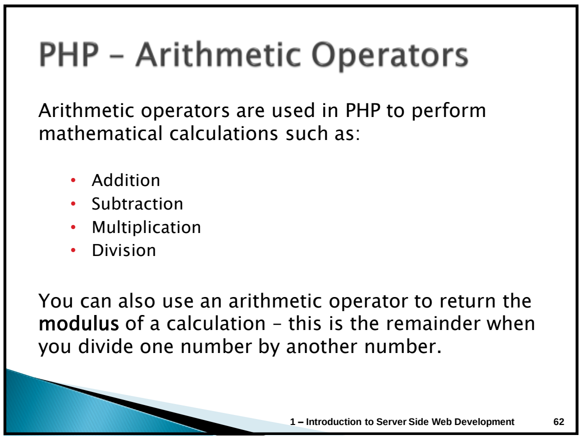
- A **binary operator** requires an operand *before* and *after* the operator

```
$MyNumber = 100;
```

- A **unary operator** requires a single operand either before or after the operator

```
$MyNumber++;
```

1 - Introduction to Server Side Web Development 61



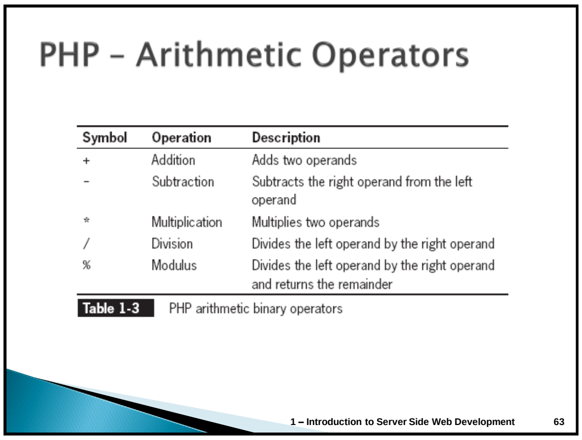
PHP – Arithmetic Operators

Arithmetic operators are used in PHP to perform mathematical calculations such as:

- Addition
- Subtraction
- Multiplication
- Division

You can also use an arithmetic operator to return the **modulus** of a calculation – this is the remainder when you divide one number by another number.

1 – Introduction to Server Side Web Development 62

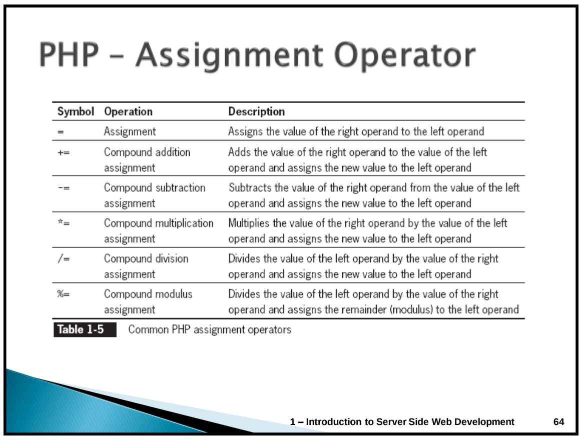


PHP – Arithmetic Operators

Symbol	Operation	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts the right operand from the left operand
*	Multiplication	Multiplies two operands
/	Division	Divides the left operand by the right operand
%	Modulus	Divides the left operand by the right operand and returns the remainder

Table 1-3 PHP arithmetic binary operators

1 – Introduction to Server Side Web Development 63



PHP – Assignment Operator

Symbol	Operation	Description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the new value to the left operand
-=	Compound subtraction assignment	Subtracts the value of the right operand from the value of the left operand and assigns the new value to the left operand
*=	Compound multiplication assignment	Multiplies the value of the right operand by the value of the left operand and assigns the new value to the left operand
/=	Compound division assignment	Divides the value of the left operand by the value of the right operand and assigns the new value to the left operand
%=	Compound modulus assignment	Divides the value of the left operand by the value of the right operand and assigns the remainder (modulus) to the left operand

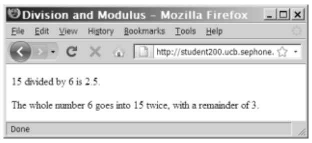
Table 1-5 Common PHP assignment operators

1 – Introduction to Server Side Web Development 64

PHP – Arithmetic Operators

Use of arithmetic operators:

```
…
$DivisionResult = 15 / 6;
$ModulusResult = 15 % 6;
echo "<p>15 divided by 6 is
    $DivisionResult.</p>"; // prints '2.5'
echo "The whole number 6 goes into 15 twice, with a
    remainder of $ModulusResult.</p>"; // prints '3'
```


Division and Modulus - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.
Done

```
15 divided by 6 is 2.5.

The whole number 6 goes into 15 twice, with a remainder of 3.
```

PHP – Unary Arithmetic Operators

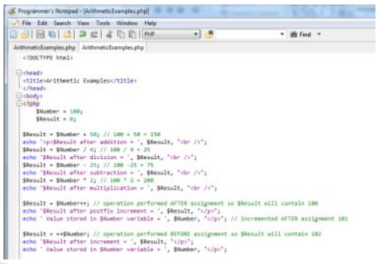
- The increment (++) and decrement (--) *unary* operators can be used as prefix or postfix operators
- A **prefix operator** is placed before a variable
- A **postfix operator** is placed after a variable

Symbol	Operation	Description
++	Increment	Increases an operand by a value of 1
--	Decrement	Decreases an operand by a value of 1

Table 1-4 PHP arithmetic unary operators

PHP – Arithmetic Operators

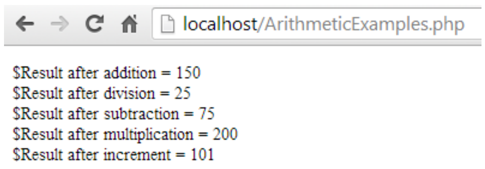
- Sample code in `ArithmeticExamples.php`:


Programmer's Notepad - [ArithmeticExamples.php]
File Edit Search View Tools Window Help
Find
ArithmeticExamples.php
ArithmeticExamples.php

```
<!DOCTYPE html>

<head>
<title>Arithmetic Examples</title>
</head>
<body>
<?php
    $Number = 100;
    $Result = 0;

$Result = $Number + 50; // 100 + 50 = 150
echo '<p>$Result after addition = ', $Result, "<br />";
$Result = $Number / 4; // 100 / 4 = 25
echo '$Result after division = ', $Result, "<br />";
$Result = $Number - 25; // 100 -25 = 75
echo '$Result after subtraction = ', $Result, "<br />";
$Result = $Number * 2; // 100 * 2 = 200
echo '$Result after multiplication = ', $Result, "<br />";

$Result = $Number++; // operation performed AFTER assignment so $Result will contain 100
echo '$Result after postfix increment = ', $Result, "</p>";
echo ' Value stored in $Number variable = ', $Number, "</p>"; // incremented AFTER assignment 101

$Result = ++$Number; // operation performed BEFORE assignment so $Result will contain 102
echo '$Result after increment = ', $Result, "</p>";
echo ' Value stored in $Number variable = ', $Number, "</p>";
```

PHP – Arithmetic Operators

Results of arithmetic expressions:



```
$Result after addition = 150
$Result after division = 25
$Result after subtraction = 75
$Result after multiplication = 200
$Result after increment = 101
```

PHP– Operands

- Expressions consist of two types of components: *operands* and *operators.*
- Operands are the objects that are manipulated and operators are the symbols that represent specific actions.

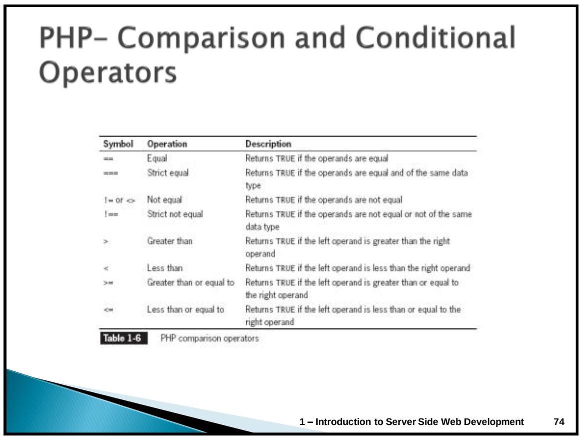
For example, in the expression

5 + x

- *x* and 5 are operands and + is an operator. All expressions have at least one operand.

PHP– Comparison and Conditional Operators

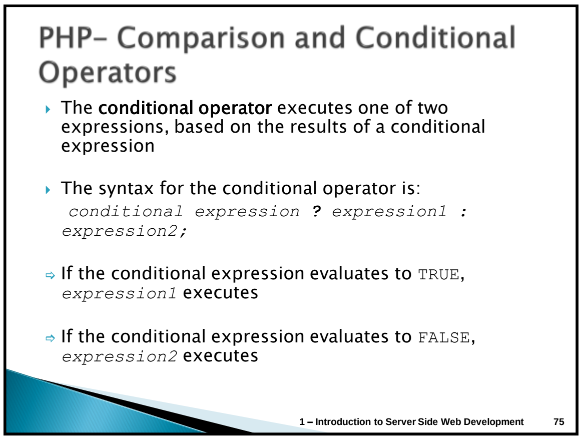
- **Comparison operators** are used to compare two operands and determine how one operand compares to another
- A Boolean value of `TRUE` or `FALSE` is returned after two operands are compared
- The comparison operator *compares* values, whereas the assignment operator *assigns* values
- Comparison operators are used with **conditional statements** and **looping statements**



PHP– Comparison and Conditional Operators

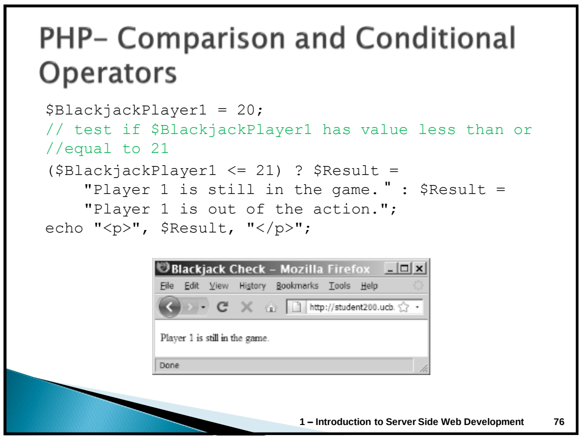
Symbol	Operation	Description
==	Equal	Returns TRUE if the operands are equal
===	Strict equal	Returns TRUE if the operands are equal and of the same data type
!= or <>	Not equal	Returns TRUE if the operands are not equal
!==	Strict not equal	Returns TRUE if the operands are not equal or not of the same data type
>	Greater than	Returns TRUE if the left operand is greater than the right operand
<	Less than	Returns TRUE if the left operand is less than the right operand
>=	Greater than or equal to	Returns TRUE if the left operand is greater than or equal to the right operand
<=	Less than or equal to	Returns TRUE if the left operand is less than or equal to the right operand

Table 1-6 PHP comparison operators



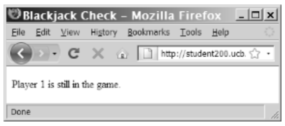
PHP– Comparison and Conditional Operators

- The **conditional operator** executes one of two expressions, based on the results of a conditional expression
- The syntax for the conditional operator is:
  `conditional expression ? expression1 : expression2;`
  - If the conditional expression evaluates to `TRUE`, *`expression1`* executes
  - If the conditional expression evaluates to `FALSE`, *`expression2`* executes

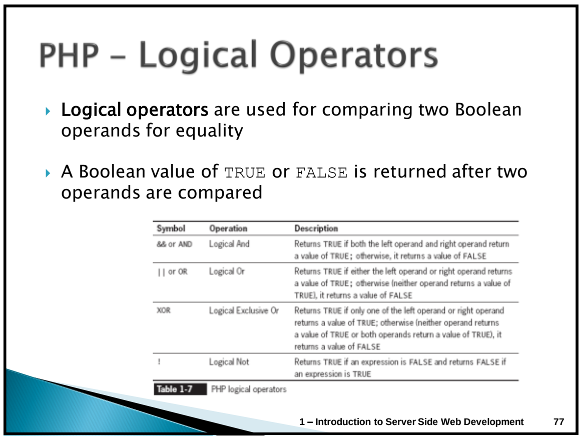


PHP– Comparison and Conditional Operators

```
$BlackjackPlayer1 = 20;
// test if $BlackjackPlayer1 has value less than or
//equal to 21
($BlackjackPlayer1 <= 21) ? $Result =
     "Player 1 is still in the game." : $Result =
     "Player 1 is out of the action.";
echo "<p>", $Result, "</p>";
```


Blackjack Check – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.
Done

Player 1 is still in the game.

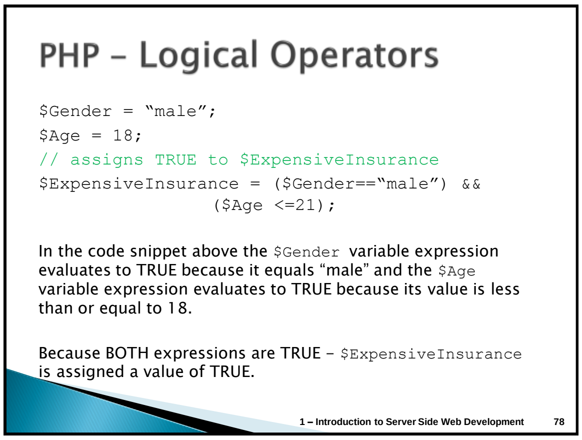


PHP – Logical Operators

- **Logical operators** are used for comparing two Boolean operands for equality
- A Boolean value of `TRUE` or `FALSE` is returned after two operands are compared

Symbol	Operation	Description
&& or AND	Logical And	Returns TRUE if both the left operand and right operand return a value of TRUE; otherwise, it returns a value of FALSE
|| or OR	Logical Or	Returns TRUE if either the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE), it returns a value of FALSE
XOR	Logical Exclusive Or	Returns TRUE if only one of the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE or both operands return a value of TRUE), it returns a value of FALSE
!	Logical Not	Returns TRUE if an expression is FALSE and returns FALSE if an expression is TRUE

Table 1-7 PHP logical operators

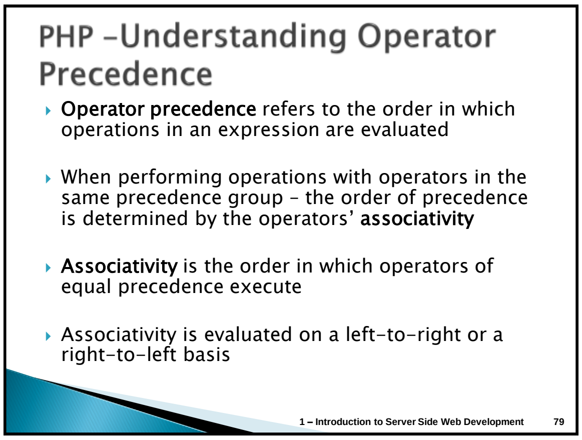


PHP – Logical Operators

```
$Gender = "male";
$Age = 18;
// assigns TRUE to $ExpensiveInsurance
$ExpensiveInsurance = ($Gender=="male") &&
                      ($Age <=21);
```

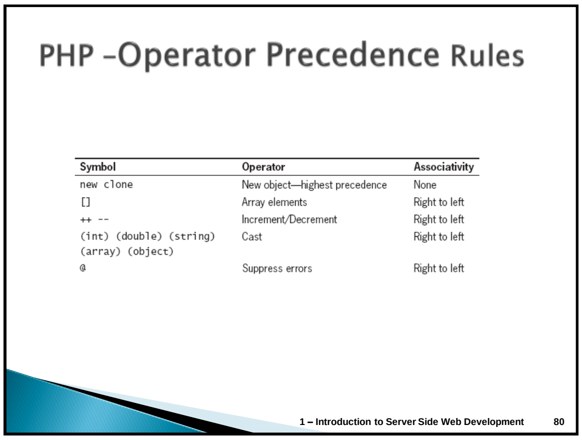
In the code snippet above the `$Gender` variable expression evaluates to TRUE because it equals "male" and the `$Age` variable expression evaluates to TRUE because its value is less than or equal to 18.

Because BOTH expressions are TRUE – `$ExpensiveInsurance` is assigned a value of TRUE.



PHP –Understanding Operator Precedence

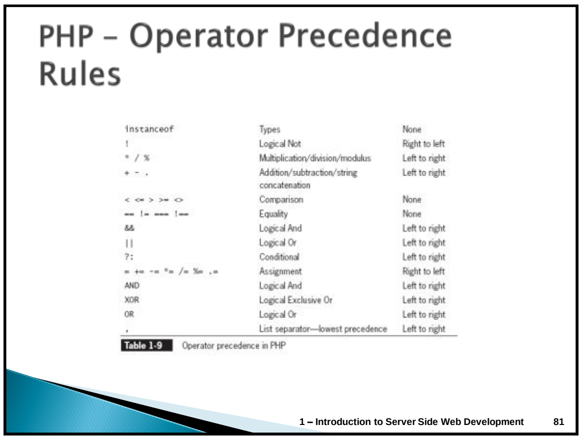
- **Operator precedence** refers to the order in which operations in an expression are evaluated
- When performing operations with operators in the same precedence group – the order of precedence is determined by the operators' **associativity**
- **Associativity** is the order in which operators of equal precedence execute
- Associativity is evaluated on a left-to-right or a right-to-left basis



PHP –Operator Precedence Rules

Symbol	Operator	Associativity
`new clone`	New object—highest precedence	None
`[]`	Array elements	Right to left
`++ --`	Increment/Decrement	Right to left
`(int) (double) (string) (array) (object)`	Cast	Right to left
`@`	Suppress errors	Right to left

1 – Introduction to Server Side Web Development 80

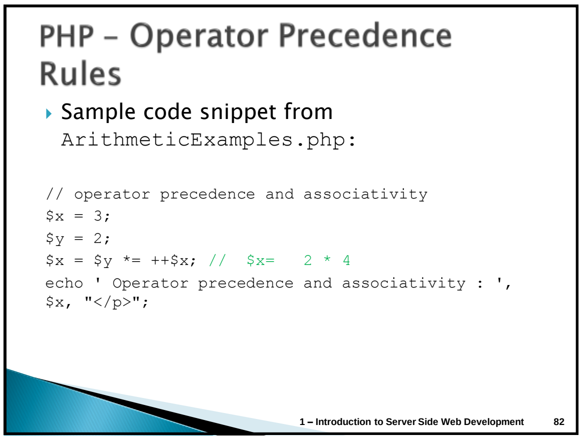


PHP – Operator Precedence Rules

`instanceof`	Types	None
`!`	Logical Not	Right to left
`* / %`	Multiplication/division/modulus	Left to right
`+ - .`	Addition/subtraction/string concatenation	Left to right
`< <= > >= <>`	Comparison	None
`== != === !==`	Equality	None
`&&`	Logical And	Left to right
`||`	Logical Or	Left to right
`?:`	Conditional	Left to right
`= += -= *= /= %= .=`	Assignment	Right to left
AND	Logical And	Left to right
XOR	Logical Exclusive Or	Left to right
OR	Logical Or	Left to right
`,`	List separator—lowest precedence	Left to right

Table 1-9 Operator precedence in PHP

1 – Introduction to Server Side Web Development 81



PHP – Operator Precedence Rules

- Sample code snippet from `ArithmeticExamples.php:`

```
// operator precedence and associativity
$x = 3;
$y = 2;
$x = $y *= ++$x; //  $x=   2 * 4
echo ' Operator precedence and associativity : ',
$x, "</p>";
```

1 – Introduction to Server Side Web Development 82