

# **Data Types and Operators (Part 2)**

**MIS 4330  
Dr. Garrett**

# 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 value such as a literal string or a number
- **Operators** are symbols (+) (\*) that are used in expressions to manipulate operands

# Building Expressions (continued)

Table 3-2 PHP Operator Types

Operator 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

# Building Expressions (continued)

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

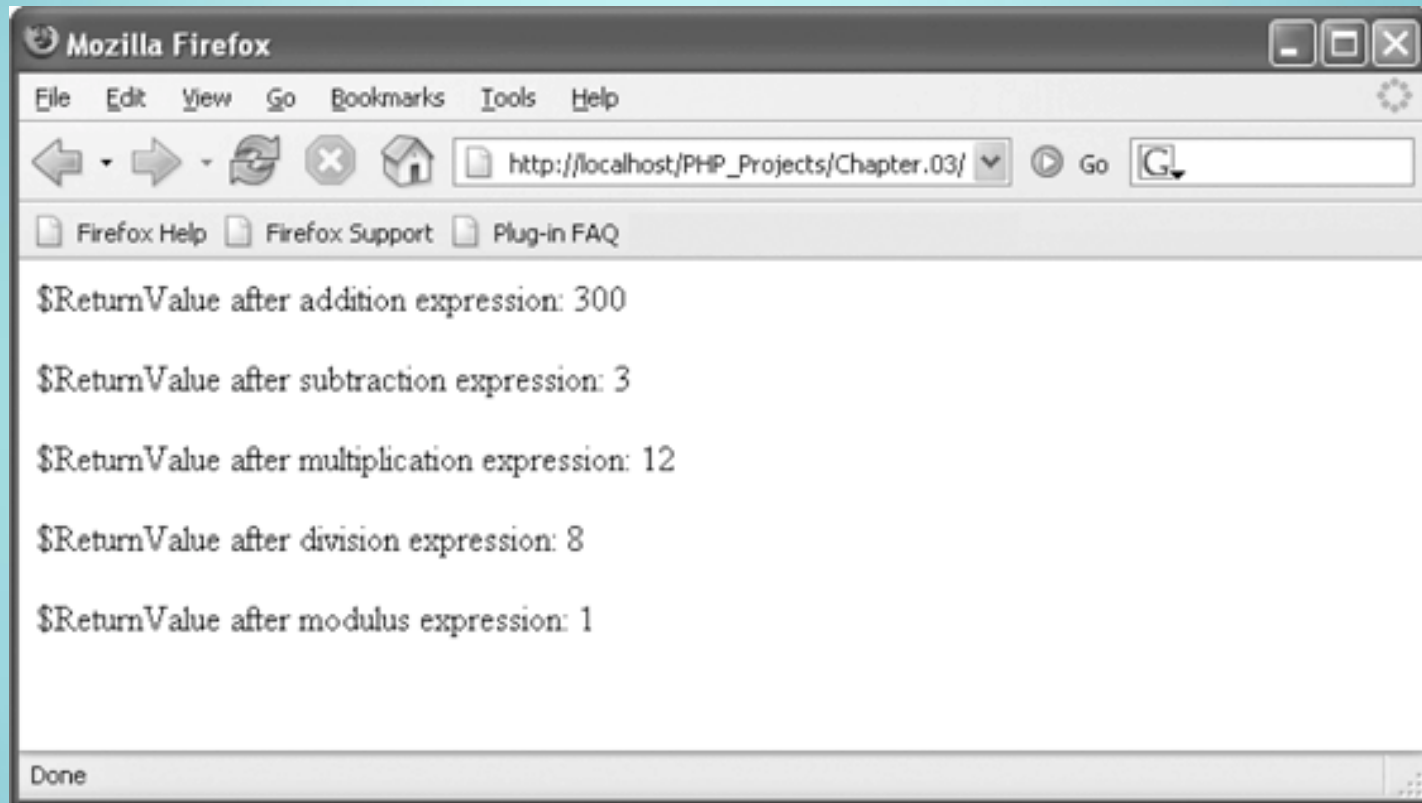
# Arithmetic Operators

- **Arithmetic operators** are used in PHP to perform mathematical calculations (+ - x ÷)

Table 3-3 PHP arithmetic binary operators

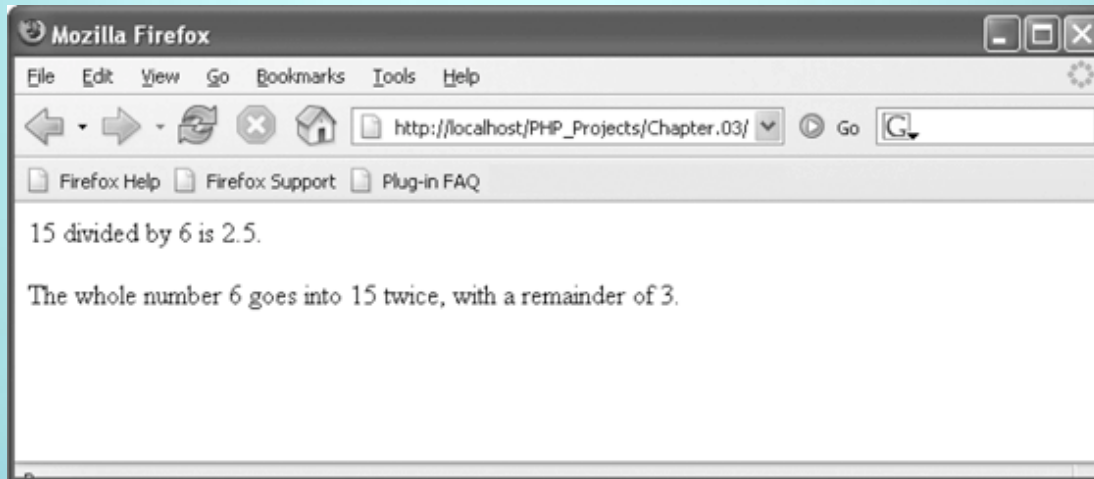
Operator	Name	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts one operand from another operand
*	Multiplication	Multiplies one operand by another operand
/	Division	Divides one operand by another operand
%	Modulus	Divides one operand by another operand and returns the remainder

# Arithmetic Operators (continued)



# Arithmetic Operators (continued)

```
$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'
```



# Arithmetic Unary 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

# Arithmetic Unary Operators (continued)

Table 3-4 PHP arithmetic unary operators

Operator	Name	Description
++	Increment	Increases an operand by a value of one
--	Decrement	Decreases an operand by a value of one

# Arithmetic Unary Operators (continued)

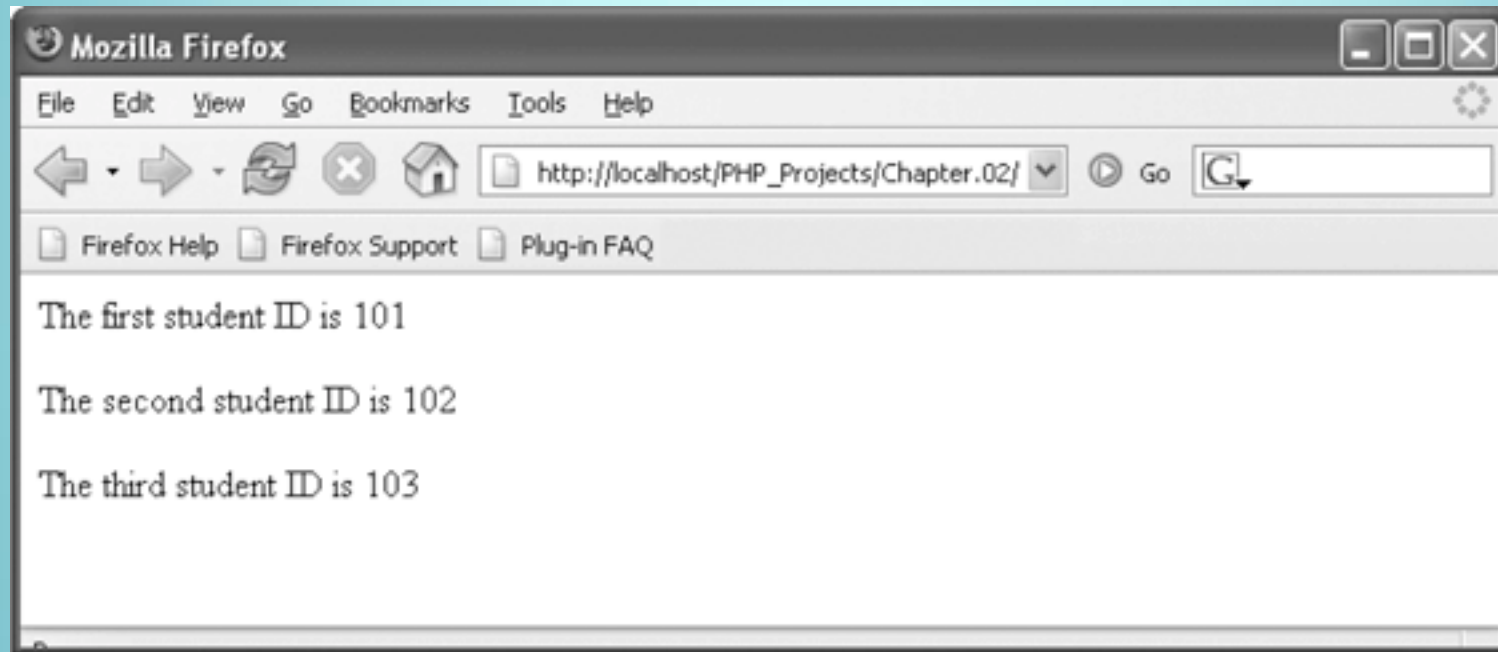
```
$StudentID = 100;  
$CurStudentID = ++$StudentID; // assigns '101'  
echo "<p>The first student ID is ",  
    $CurStudentID, "</p>";  
$CurStudentID = ++$StudentID; // assigns '102'  
echo "<p>The second student ID is ",  
    $CurStudentID, "</p>";  
$CurStudentID = ++$StudentID; // assigns '103'  
echo "<p>The third student ID is ",  
    $CurStudentID, "</p>";
```

prefix increment operator

**Figure 3-14 Script that uses the prefix increment operator**

# Arithmetic Unary Operators (continued)

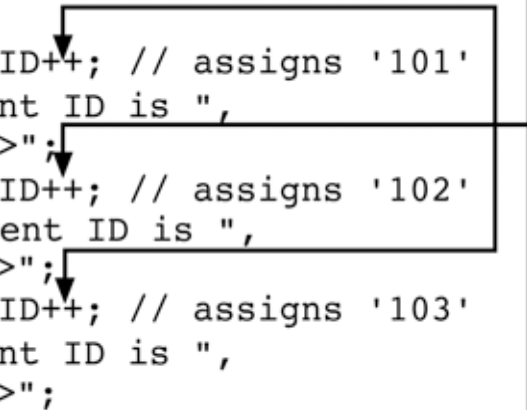
Figure 3-15 Output of the prefix version of the student ID script



# Arithmetic Unary Operators (continued)

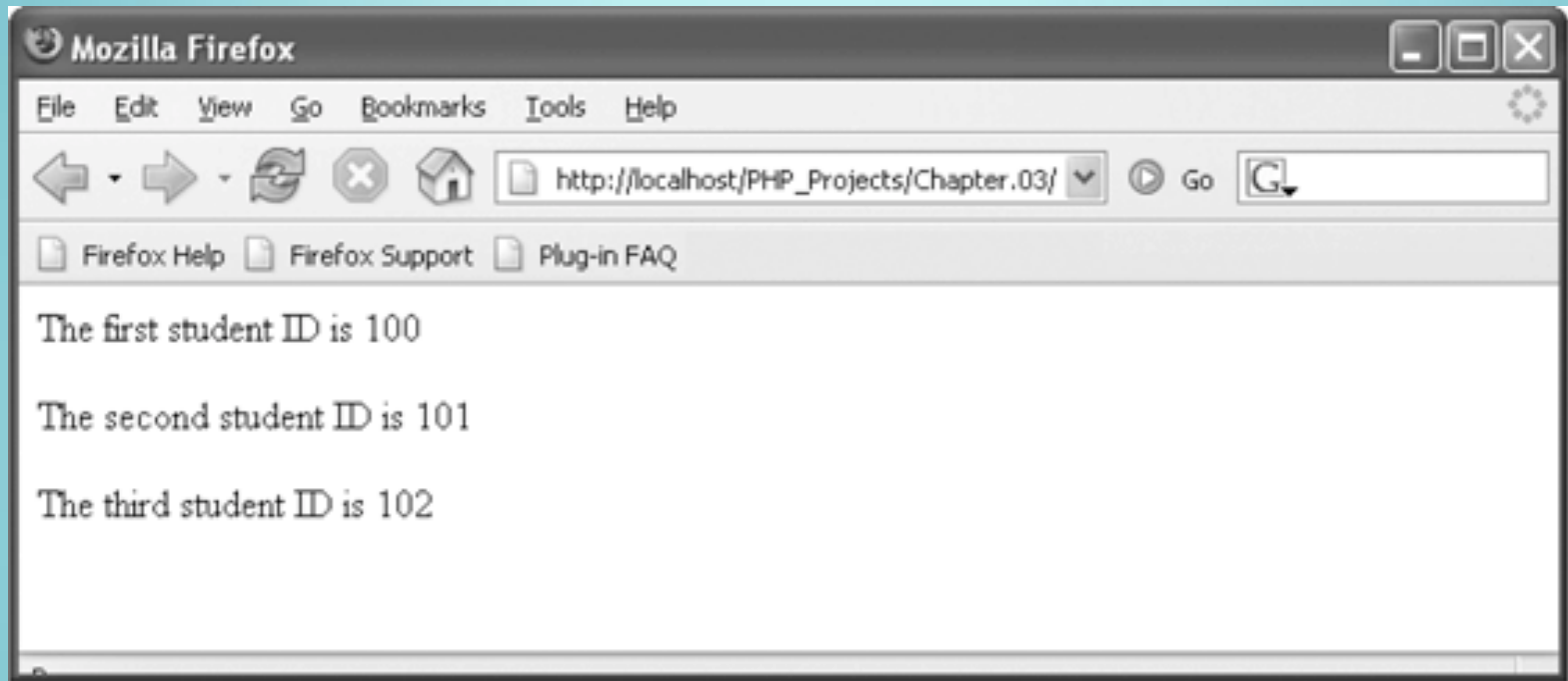
```
$StudentID = 100;  
$CurStudentID = $StudentID++; // assigns '101'  
echo "<p>The first student ID is ",  
    $CurStudentID, "</p>";  
$CurStudentID = $StudentID++; // assigns '102'  
echo "<p>The second student ID is ",  
    $CurStudentID, "</p>";  
$CurStudentID = $StudentID++; // assigns '103'  
echo "<p>The third student ID is ",  
    $CurStudentID, "</p>";
```

postfix increment operator



**Figure 3-16 Script that uses the postfix increment operator**

# Arithmetic Unary Operators (continued)



**Figure 3-17 Output of the postfix version of the student ID script**

# Assignment Operators

- **Assignment operators** are used for assigning a value to a variable:

```
$MyFavoriteSuperHero = "Superman";  
$MyFavoriteSuperHero = "Batman";
```

- **Compound assignment operators** perform mathematical calculations on variables and literal values in an expression, and then assign a new value to the left operand

# Assignment Operators (continued)

Table 3-5 PHP assignment operators

Operator	Name	Description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Combines the value of the right operand with the value of the left operand or 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

# 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**

# Comparison and Conditional Operators (continued)

Table 3-6 PHP comparison operators

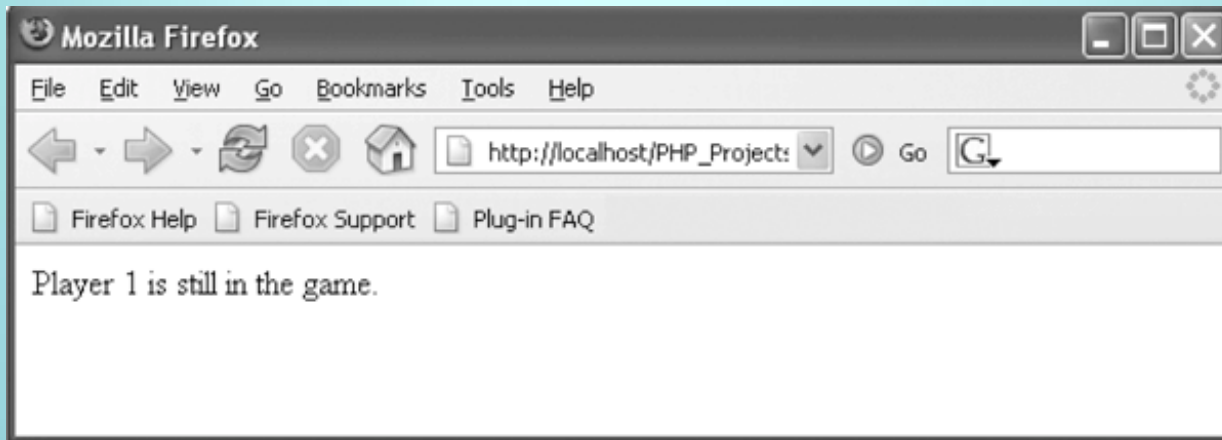
Operator	Name	Description
==	Equal	Returns true if the operands are equal
===	Strict equal	Returns true if the operands are equal and of the same type
!= or <>	Not equal	Returns true if the operands are not equal
!==	Strict not equal	Returns true if the operands are not equal or not of the same type
>	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

# Comparison and Conditional Operators (continued)

- 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

# Comparison and Conditional Operators (continued)

```
$BlackjackPlayer1 = 20;  
$BlackjackPlayer1 <= 21) ? $Result =  
    "Player 1 is still in the game." :  
$Result = "Player 1 is out of the action.";  
echo "<p>", $Result, "</p>"
```



**Figure 3-21** Output of a script with a conditional operator

# 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

Table 3-7 PHP logical operators

Operator	Name	Description
&&, and	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	Returns true if either the left operand or right operand returns a value of true; if neither operand returns a value of true the expression containing the Or (   ) operator returns a value of false
!	Not	Returns true if an expression is false and returns false if an expression is true

# Special Operators

Table 3-8 PHP special operators

Operator	Description
<code>new</code>	Creates a new instance of a user-defined object type or a predefined PHP object type
<code>[ ]</code>	Accesses an element of an array
<code>=&gt;</code>	Specifies the index or key of an array element
<code>,</code>	Separates arguments in a list
<code>?:</code>	Executes one of two expressions based on the results of a conditional expression
<code>instanceof</code>	Returns true if an object is of a specified object type
<code>@</code>	Suppresses any errors that might be generated by an expression to which it is prepended (or "placed before")
<code>(int), (integer), (bool), (boolean), (double), (string), (array), (object)</code>	Casts (or transforms) a variable of one data type into a variable of another data type

# Type Casting

- **Casting** or **type casting** copies the value contained in a variable of one data type into a variable of another data type
- The PHP syntax for casting variables is:  
`$NewVariable = (new_type) $OldVariable;`
- `(new_type)` refers to the type-casting operator representing the type to which you want to cast the variable

# gettype () function

- Returns one of the following strings, depending on the data type:
  - Boolean
  - Integer
  - Double
  - String
  - Array
  - Object
  - Resource
  - NULL
  - Unknown type

# Understanding Operator Precedence

- **Operator precedence** refers to the order in which operations in an expression are evaluated
- **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

# Understanding Operator Precedence (continued)

**Table 3-9 Operator precedence in PHP**

Operators	Description	Assodativty
<code>new</code>	New object—highest precedence	None
<code>[]</code>	Array elements	Right to left
<code>!</code>	Logical Not	Right to left
<code>++</code>	Increment	Right to left
<code>--</code>	Decrement	Right to left
<code>(int), (double), (string), (array), (object)</code>	Cast	Right to left
<code>@</code>	Suppress errors	Right to left
<code>* / %</code>	Multiplication/division/modulus	Left to right
<code>+ - .</code>	Addition/subtraction/string concatenation	Left to right
<code>&lt; &lt;= &gt; &gt;=</code>	Comparison	None
<code>== != &lt;&gt; === !==</code>	Equality	None
<code>&amp;&amp;</code>	Logical And	Left to right
<code>  </code>	Logical Or	Left to right
<code>?:</code>	Conditional	Left to right
<code>= += -= *= /= %=</code>	Assignment	Right to left
<code>and</code>	Logical And	Left to right
<code>or</code>	Logical Or	Left to right
<code>,</code>	List separator—lowest precedence	Left to right