SWINBURNE UNIVERSITY OF TECHNOLOGY

Advanced Web Development: Data Types and Operators

Week 2

Outline

- Working with Data Types and Operators
- Work with variables and constants
- Study data types
- Use expressions and operators
- Cast the data types of variables
- Learn about operator precedence

- Reading: Textbook Chapter 1

Using Variables and Constants

- The values stored in computer memory are called **variables**
- The values, or data, contained in variables are classified into categories known as **data types**
- The name you assign to a variable is called an **identifier** and it:
  - Must begin with a dollar sign ($)
  - Can include letters (A to Z, a to z) and numbers (0 to 9) or an underscore (_) … but cannot start with a number
  - Cannot include spaces
  - Is case sensitive

Naming Variables

- You must follow a consistent variable naming style
  - `$votingAge`
  - `$voting_age`
  - `$votingage`
  - `$VotingAge`
  - `$VOTING_AGE`
- Are the two variable names below referring to the same variable (identifier)?
  - `$firstName`
  - `$FirstName`

Declaring, Initialising and Modifying Variables

- Specifying and creating a variable name is called **declaring the variable**
- Assigning a first value to a variable is called **initialising the variable**
- In PHP, you must declare and <u>initialise</u> a variable in the same statement:

```
$variable_name = value;
```

- You can change the variable's value at any point

```
$variable_name = new_value;
```

Displaying the Values of Variables

- To print a variable with the `echo` statement, pass the variable name to the `echo` statement without enclosing it in quotation marks:

```
 $votingAge = 18;

 echo $votingAge;
```

- To print both text strings and variables, send them to the `echo` statement as individual arguments, separated by commas:

```
echo "<p>The legal voting age is ",
   $votingAge, ".</p>";
```

Displaying the Values of Variables

- What if surrounded by double/single quotation marks

```
echo "<p>The legal voting age is
        $votingAge.</p>";
```

Content of $votingAge will be printed

```
echo '<p>The legal voting age is
        $votingAge.</p>';
```

Text '$votingAge' itself will be printed out.

Hint: If in doubt, separate with commas

```
echo "<p>The legal voting age is ",
        $votingAge, ".</p>”;
```

Defining Constants

- A constant contains information that does not change during the course of program execution
- Constant names do not begin with a dollar sign ($)
- Constant names use all uppercase letters
- Use the **define()** function to create a constant
  - `define("CONSTANT_NAME", value);`
- The value you pass to the define() function can be a text string, number, or Boolean value
- PHP includes numerous predefined constants that you can use in your scripts

Naming Constants

- You must follow a consistent constant naming style
  - ☐ `PASSING_MARK`
  - ☐ `PASINGMARK`
  - ☐ `Passing_Mark`
  - ☐ `passing_mark`
  - ☐ `passingMark`
- Which one of the following is a constant?
  - ☐ `$MAX_ELEMENTS`
  - ☐ `MAX_ELEMENTS`

Working with Data Types

- A **data type** is the specific category of information that a variable contains
- Data types that can be assigned only a single value are called **primitive types**

Primitive PHP data types

Data Type	Description
Integer numbers	Positive or negative numbers 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

Working with Data Types (continued)

- **Strongly typed programming languages** require you to declare the data types of variables
  - **Static** or **strong typing** refers to data types that ***do not*** change after they have been declared
  - C is a strongly typed programming language
- **Loosely typed programming languages** do not require you to declare the data types of variables
  - **Dynamic** or **loose typing** refers to data types that can change after they have been declared
  - PHP is a loosely typed programming language

Numeric Data Types

PHP supports two numeric data types:

- An **integer** is a positive or negative number 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 short for writing very large numbers or numbers with many decimal places (2.0e11)

Boolean Values

- A **Boolean value** is a value of true or false
- It decides which part of a program should execute and which part should compare data
- In PHP programming, you can only use true or false
- In other programming languages, you can use integers such as 1 = true, 0 = false

Working with Data Types (continued)

- The data type of a variable (identifiers) or constant depends on the data type of the value assigned to it
  - ☐ $unitName = “Web Programming”;
  - ☐ $lectureHours = 2;
  - ☐ $creditPoints = 12.5;
  - ☐ $isCoreUnit = TRUE;

Hint: Give meaningful names

Did you notice any naming pattern?

- Are the following correct?
  - ☐ $unitCode = COS30020;
  - ☐ $creditPoints = 12.5cp;

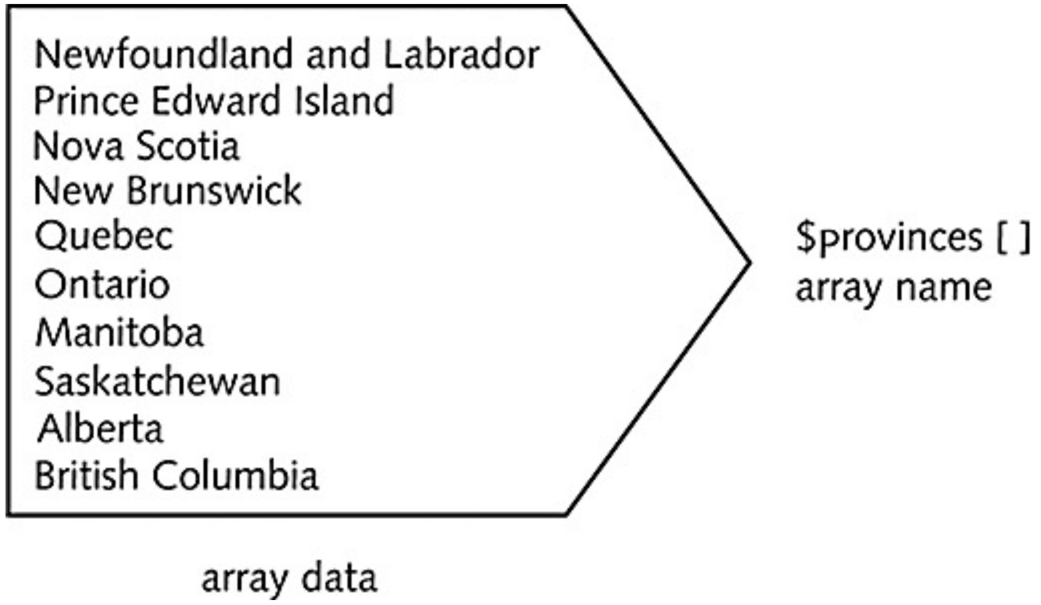
Note: COS30020 may be a constant

Working with Data Types (continued)

- The PHP language 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**

Arrays

- An **array** contains a set of data represented by a single variable name



Conceptual example of an array

- Indexed arrays & associative arrays (Chapter 6)

Declaring and Initialising Indexed Arrays

- An **element** refers to each piece of data that is stored within an array
  - By default, it starts with the number zero (0)
- An **index** is an element's numeric position within the array
  - Referenced by enclosing its index in brackets at the end of the array name:
  - `$provinces[1]`

Creating an Array

- The `array()` construct syntax is:

```
$array_name = array(values);
```

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

Creating an Array (continued)

- Array name and brackets syntax is:

    ***$array_name*[ ]**

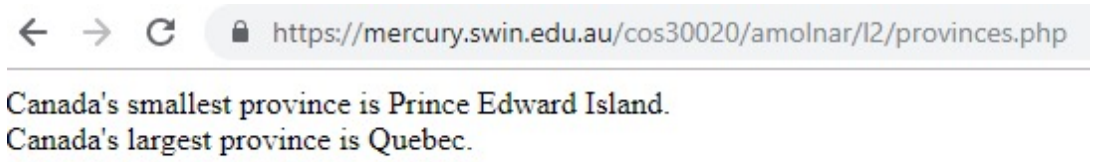
```
$provinces[] = "Newfoundland and Labrador";
$provinces[] = "Prince Edward Island";
$provinces[] = "Nova Scotia";
$provinces[] = "New Brunswick";
$provinces[] = "Quebec";
$provinces[] = "Ontario";
$provinces[] = "Manitoba";
$provinces[] = "Saskatchewan";
$provinces[] = "Alberta";
$provinces[] = "British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

***Note:** In PHP, array elements can be of different data types*

Accessing Element Information

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

-
  ```
  echo "Canada's largest province is
      $provinces[4].</p>";
  ```



```
https://mercury.swin.edu.au/cos30020/amolnar/l2/provinces.php

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

Output of elements in the $provinces[ ] array

`count()` Function

- 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>";
```

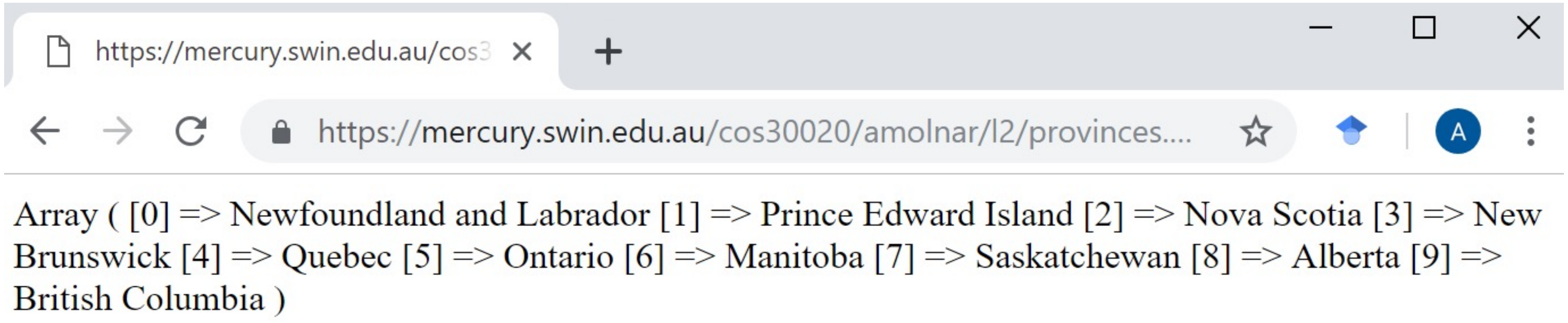
Output:

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

SWINBURNE UNIVERSITY OF TECHNOLOGY

`print_r()` Function

- Use to print or return information about variables
- Most useful with arrays because they print the index and value of each element


https://mercury.swin.edu.au/cos3
https://mercury.swin.edu.au/cos30020/amolnar/l2/provinces....

```
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 )
```

Output of the `$provinces[]` array with the `print_r()` function

Modifying Elements

- 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";
```

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 (e.g. +, *) that are used in expressions to manipulate operands

Building Expressions (continued)

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 the operator categories
String	Performs operations on strings

Note: Details about Arrays are in Chapter 6 and Strings are in Chapter 3

- 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

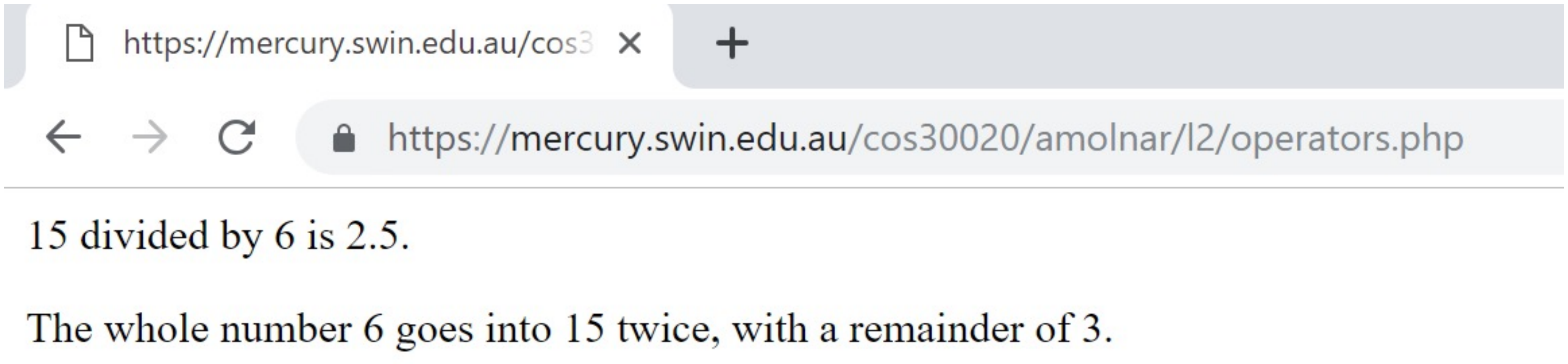
- **Arithmetic operators** are used in PHP to perform mathematical calculations

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 reminder

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'
```


https://mercury.swin.edu.au/cos3
https://mercury.swin.edu.au/cos30020/amolnar/l2/operators.php

```
15 divided by 6 is 2.5.

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

Division and modulus expressions

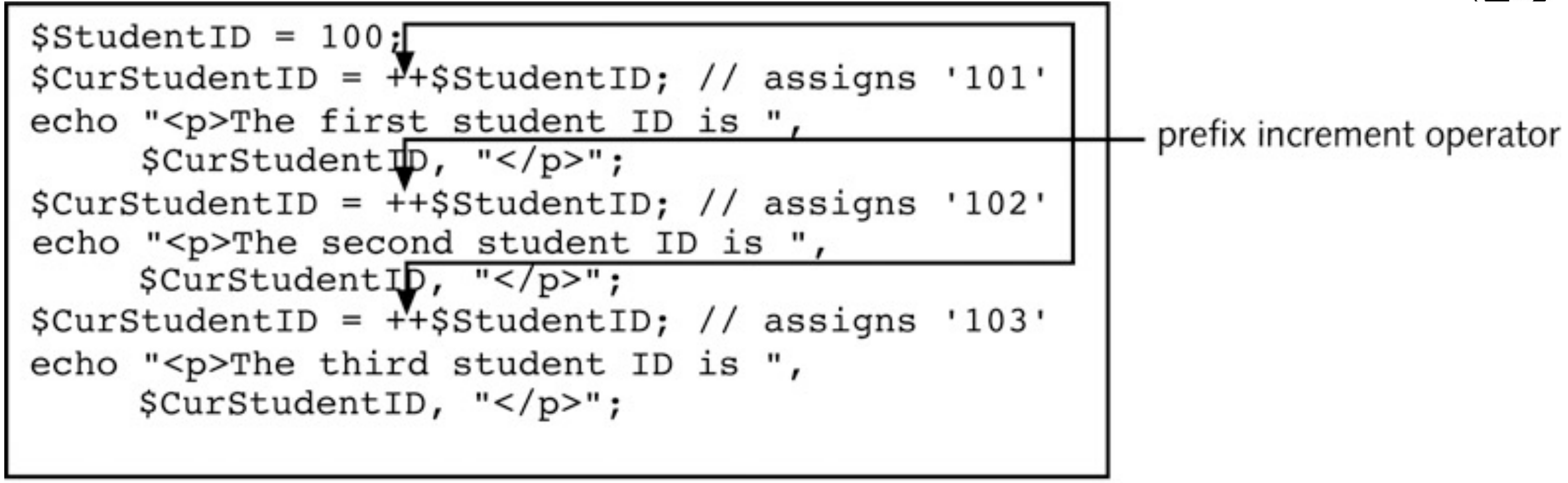
Arithmetic Unary Operators

- 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

PHP arithmetic unary operators

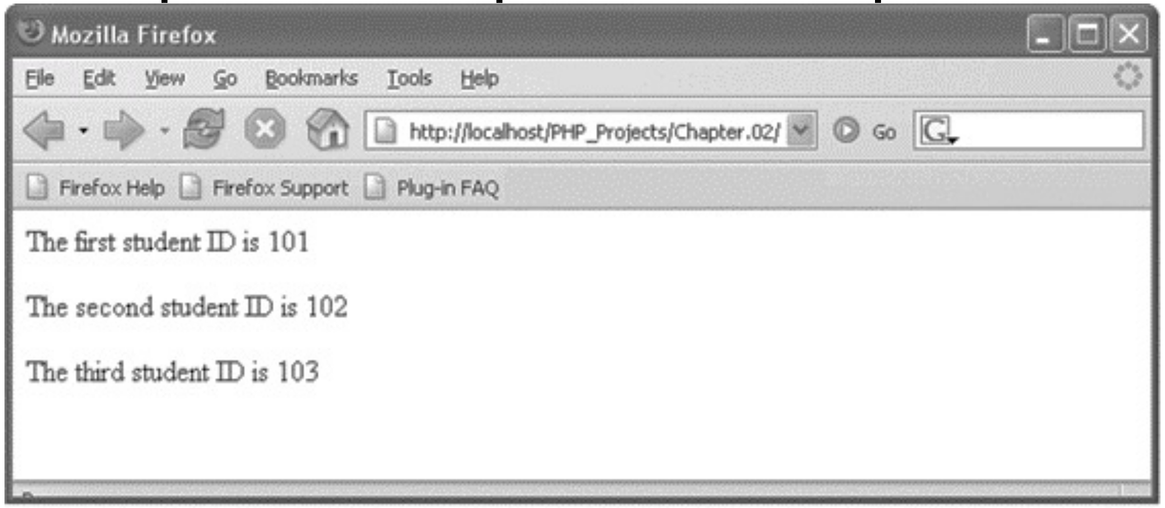
Operator	Name	Description
++	Increment	Increases an operand by a value of one
--	Decrement	Decreases an operand by a value of one

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>";
```

Script that uses the prefix increment operator



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.02/
Go
Firefox Help
Firefox Support
Plug-in FAQ

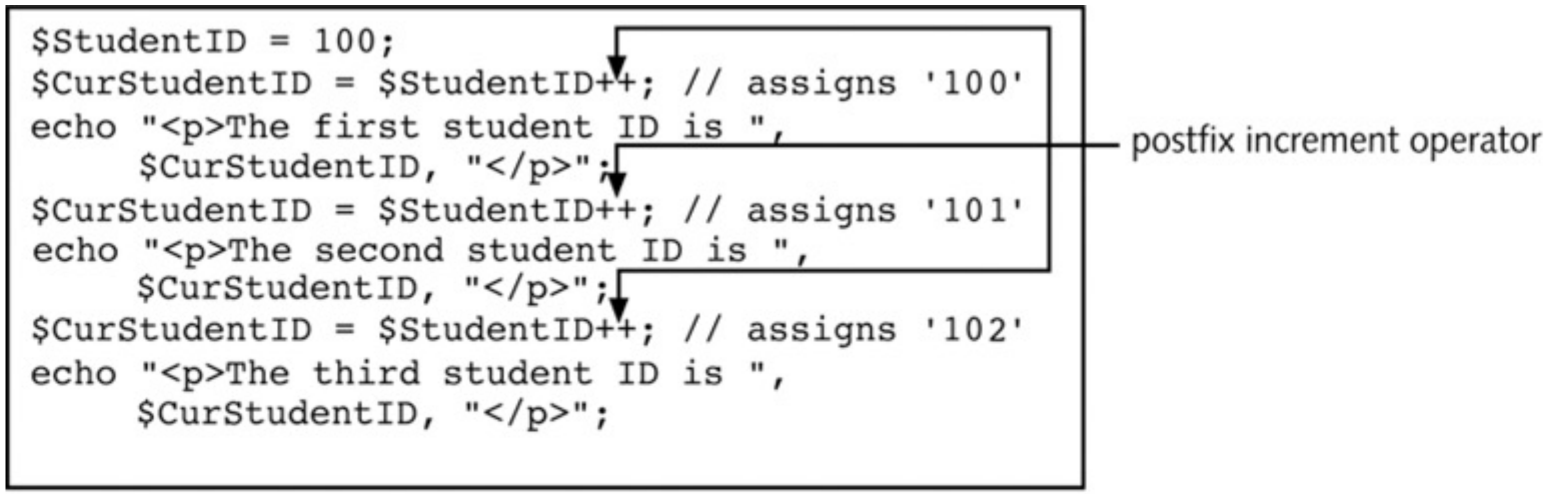
The first student ID is 101

The second student ID is 102

The third student ID is 103

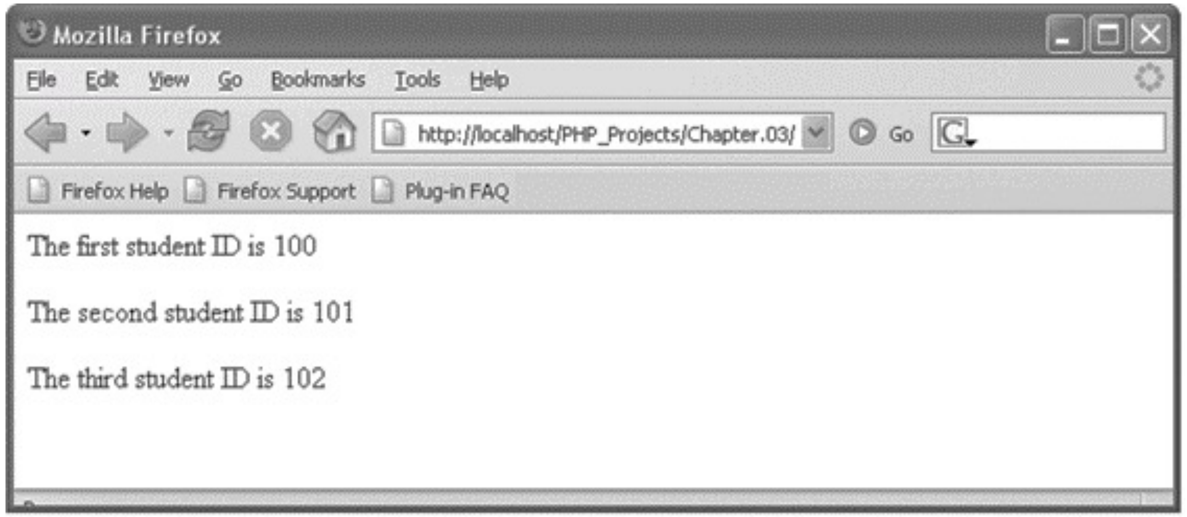
Output of the prefix version of the student ID script

Arithmetic Unary Operators (continued)



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

Script that uses the postfix increment operator



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.03/
Go
Firefox Help
Firefox Support
Plug-in FAQ

```
The first student ID is 100

The second student ID is 101

The third student ID is 102
```

Output of the postfix version of the student ID script

Arithmetic Unary Operators (continued)

- What is the difference between prefix increment operator and postfix increment operator ?

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)

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 (modulus) to the left operand

Assignment Operators (continued)

```
$x = 100;
$y = 200;
$x += $y;
```
same as `$x = $x + $y;` (300)

```
$x = 2;
$y = 6;
$x *= $y;
```
same as `$x = $x * $y;` (12)

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)

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

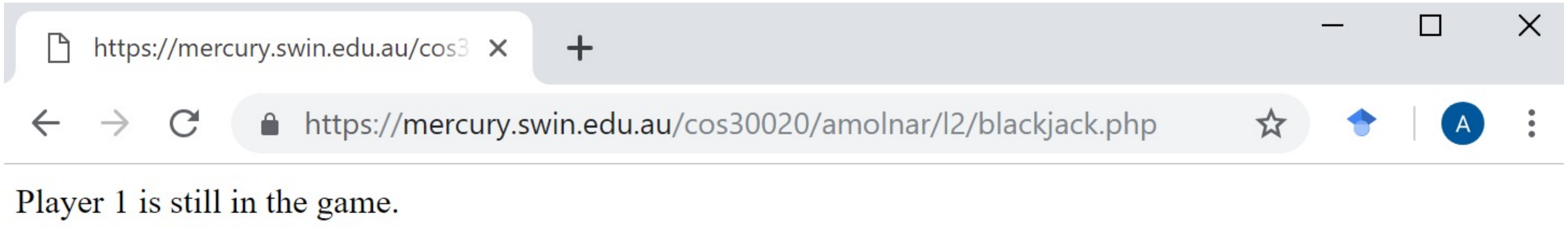
Comparison and Conditional Operators

(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>";
```


https://mercury.swin.edu.au/cos30020/amolnar/l2/blackjack.php

Player 1 is still in the game.

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

PHP logical operators

Operator	Name	Description
&&, and	And	Returns true if both the left operand and the 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 the 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

PHP special operators

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

***Note**: These Special Operators are introduced throughout this unit as necessary*

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

```
$speedLimitMiles = "55 mph";
$speedLimitKilometers = (int) $speedLimitMiles * 1.6;
echo "$speedLimitMiles is equal to
  $speedLimitKilometers kph";
```

`gettype()` function

Returns one of the following strings, depending on the data type
- *no guessing needed:*
  - ☐ 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
- What to do if not certain when you write code?

Understanding Operator Precedence

(continued)

Operator precedence in PHP

Operators	Description	Associativity
new	New object – highest precedence	None
[ ]	Array elements	Right to left
! ++ -- (int), (double), (string), (array), (object) @	Logical Not Increment Decrement Cast Supress errors	Right to left Right to left Right to left Right to left 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
or	Logical Or	Left to right
,	List separator – lowest precedence	Left to right

Summary

- The values a program stores in computer memory are called variables
- A data type is the specific category of information that a variable contains
- PHP is a loosely typed programming language
- An integer is a positive or negative number with no decimal places
- A Boolean value is a logical value of true or false

Summary (continued)

- An array contains a set of data represented by a single variable name
- Operands are variables and literals contained in an expression
- A binary operator requires an operand before and after the operator
- A unary operator requires a single operand either before or after the operator
- Assignment operators are used for assigning a value to a variable

Summary (continued)

- The conditional operator executes one of two expressions, based on the results of a conditional expression
- Logical operators are used for comparing two Boolean operands for equality
- Casting or type casting copies the value contained in a variable of one data type into a variable of another data type
- Operator precedence is the order in which operations in an expression are evaluated