ISIT307- WEB SERVER PROGRAMMING

LECTURE 1.1 - INTRODUCTION

LECTURER/COORDINATOR

Lecturer

- Dr. Elena Vlahu-Gjorgievska
    - e-mail: elenavg@uow.edu.au

Tutor

- Poh Kok Loo
    - E-mail: lubg9@outlook.com

LECTURE PLAN

- What you need to know about lectures, labs, assignments and exams
- Getting started with PHP

SUBJECT STRUCTURE

- Lectures
- Labs
- Assignments : 2 assignments

ASSESSMENT

Assessment Items	Percentage of Final Mark		Due Date
	Marks for the Item	Minimum required for a pass	
Assignment 1 (group work – 2 students)	25	N/A	As scheduled
Assignment 2 (individual work)	25		As scheduled
Final Exam (TBA)	50	20	Exam week as per schedule
Total	**100**	**50**	**The mark must be ≥50 to pass the subject**

LECTURES

- The lectures will introduce fundamental concepts and the principles of web server programming.
- The lectures will contain a sufficient number of examples to facilitate explanation of complex technical aspects.
- It is highly recommended that you implement all examples, compile and run the programs on your computer.
- I’m encouraging you to actively participate in the lecture sessions answering questions and making your own notes that will help you to better understand the material.

TUTORIALS

- Students are expected to complete the tasks during a supervised tutorial session. If more time is required to complete all exercises, this can be done before or after the tutorials.
- During the scheduled tutorials, the assignments 1 and 2 need to be presented to the tutor and short Moodle quizzes needs to be completed.

ADDITIONAL MATERIALS

- Additional materials (websites, readings and videos) can be used.
- It is a good practice for you to read/watch/implement the examples from these materials.

ASSIGNMENTS

- There will be two assignments.
- When an assignment is released, download the assignment description from the subject web site. Read carefully the specifications. Make sure you understand the requirements.
- Your solutions must be submitted electronically via the subject web site (Moodle). No submission via email will be accepted.
- Late assignments will not be accepted without a granted academic consideration.
- During the scheduled tutorials, the assignments 1 and 2 need to be presented to the tutor and short Moodle quizzes needs to be completed. The tutor will assess your solution and give you a mark according to the quality of your solution and the level of your understanding.

ASSIGNMENTS

- Assignment 1 (group work - 2 students): marks for the solution, marks for the presentation, Quiz 1 (individual) marks
  - Due: TBA (Moodle submission), presentation + quiz (TBA)

- Assignment 2 (individual work): marks for the solution, marks for the presentation, Quiz 2 (individual) marks
  - Due: TBA (Moodle submission), presentation + quiz (TBA)

**Dates are subject to changes (with the lecturer/tutor permission)*

ASSIGNMENTS

- When you submit an assessment task, you are declaring the following:
  - It is your own work and you have not copied anything from others and you have not discussed your work with others.
  - You have not plagiarised from published work (including various internet sources).
  - You have read your responsibilities under the UOW's policy on plagiarism and you understand possible consequences.
  - You have not used storage devices which can be accessed by others without passwords.
- Plagiarism = Big problems
- You may be asked to have a formal meeting with the lecturer to explain your assignment solution if there are doubts that you worked on your assignment yourself.

SUBJECT WEB SITE

- All important notices related to the subjects will be posted on the subject's Moodle site.
- Check it frequently!
- **Note: For any information published on the subject's website, it is considered that all students have been notified!**

SELF-DIRECTED STUDY

- **Listening passively is useless!**
- Attend all lectures. Take your own notes and add your own comments or questions during the lectures.
- Read/implement all materials/solutions posted on the subject web site.
- Implement examples discussed at lectures or developed during labs.
- If you have any questions, discuss it with me.

SUBJECT MATERIALS

- Recommended books:
  - Gosselin, D., Kokoska, D. and Easterbrooks, R., 2011. *PHP Programming with MySQL - The Web Technologies Series (2nd edition)*. Cengage Learning.
  - Sommerfeld, R 2024, *Unlock PHP 8: From Basic to Advanced: The next-level PHP 8 guide for dynamic web development (English Edition)*. BPB Publications, London.
  - White III, E. and Eisenhamer, J.D., 2007. *PHP 5 in Practice*. Pearson Education. (Available in UOW Library as ebook)
  - Nixon, R., 2015. *Learning PHP, MySQL, and JavaScript: A Step-By-Step Guide to Creating Dynamic Websites (Animal Guide)*. O'Reilly. (Available in UOW Library as ebook)
  - Connolly, R., 2015. *Fundamentals of web development*. Pearson Education.
  - Rahman, M., 2017. *PHP 7 Data Structures and Algorithms*. Packt Publishing Ltd.
  - Nixon, R., 2021. Learning PHP, MySQL & JavaScript. O'Reilly Media.
- Lecture notes & Labs:
  - The lecture notes are available on the subject web site (The lecture notes may not include some examples and explanations given in lectures).
  - The Labs exercises are available on the subject web site.
- Additional materials
  - http://www.w3schools.com
  - http://php.net/
  - Additional materials may be posted on the subject web site.

SOFTWARE REQUIREMENTS

- Notepad++ (Windows); Sublime Text, BBEdit, TextMate (Mac)
- WAMP Server (it comes with)
  - Apache server
  - PHP v.8.1+
  - MySQL
- Instead of WAMP can be used MAMP, LAMP or XAMPP

OBJECTIVES

- Create PHP scripts
- Create PHP code blocks
- Work with variables and constants
- Study data types
- Use expressions and operators

PHP

- PHP (PHP: Hypertext Preprocessor) is a server scripting language, and a powerful tool for making dynamic and interactive Web pages
- PHP is a widely-used open source scripting language (free)
- PHP scripts are executed on the server
- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.) and is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is easy to learn and runs efficiently on the server side

PHP

- **Embedded language** refers to code that is embedded within a Web page (HTML document)
- PHP code is typed directly into a Web page as a separate section
- A Web page containing PHP code must be saved with an extension of .php to be processed by the scripting engine
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP code is never sent to a client's Web browser; only the output of the processing is sent to the browser

PHP

- The Web page generated from the PHP code, and (X)HTML elements found within the PHP file, is returned to the client
- A file that does not contain any PHP code should be saved with an **.html** extension
- .php is the default extension that most Web servers use to process PHP scripts

CREATING PHP CODE BLOCKS

- **Code declaration blocks** are separate sections on a Web page that are interpreted by the scripting engine
- There are four types of code declaration blocks:
  - Standard PHP script delimiters
  - Short PHP script delimiters – can be disabled in php.ini configuration file
  - The `<script>` element - not supported/removed in PHP7
  - ASP-style script delimiters (<% > & <%= >) - not supported/removed in PHP7

STANDARD PHP SCRIPT DELIMITERS

- A **delimiter** is a character or sequence of characters used to mark the beginning and end of a code segment
- The standard method of writing PHP code declaration blocks is to use the **`<?php`** and **`?>`** script delimiters
- The individual lines of code that make up a PHP script are called **statements**

SHORT PHP SCRIPT DELIMITERS

- The syntax for the short PHP script delimiters is

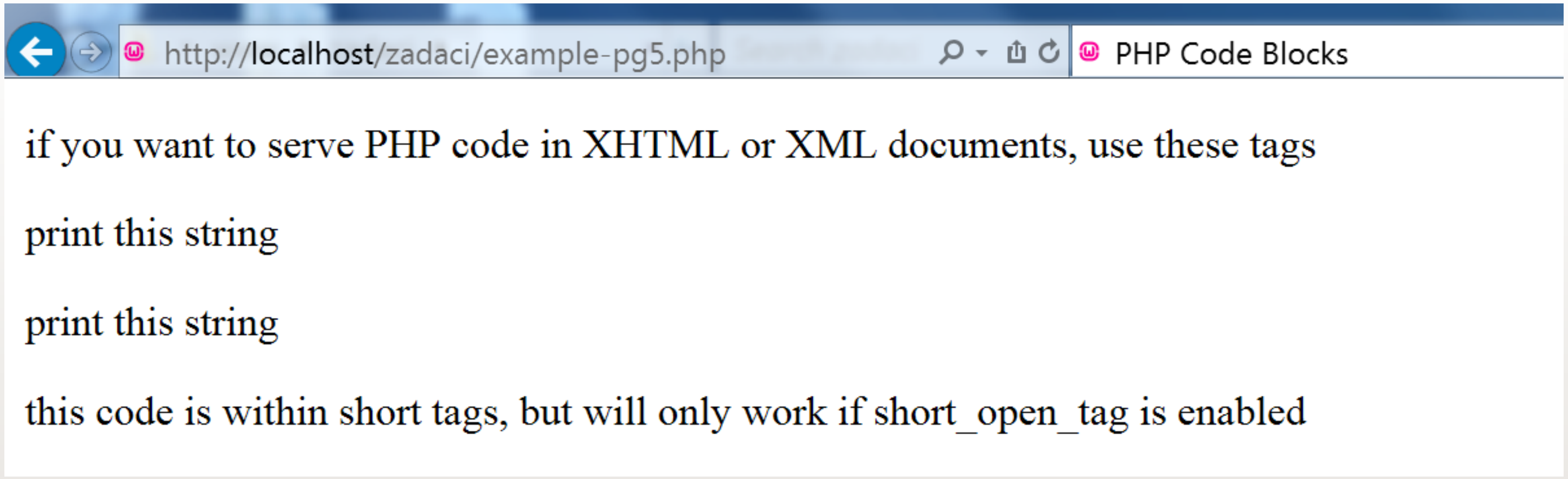
```
<? statements; ?>
```

- Short delimiters can be disabled in a Web server's php.ini configuration file
- PHP scripts will not work if your Web site ISP does not support short PHP script delimiters
- Short delimiters can be used in HTML documents, but not in XML documents

PHP SCRIPT DELIMITERS EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
        <title>PHP Code Blocks</title>
        <meta charset="utf-8" />
</head>
<body>
<p>
<?php echo 'if you want to serve PHP code in HTML or XML documents, use these tags'; ?>
</p>
<p>
<?php echo 'print this string' ?>
</p>
<p>
<?= 'print this string' ?>
</p>
<p>
<? echo 'this code is within short tags, but will only work if short_open_tag is
enabled'; ?>
</p>
</body>
</html>
```

PHP SCRIPT DELIMITERS EXAMPLE - OUTPUT



```
if you want to serve PHP code in XHTML or XML documents, use these tags

print this string

print this string

this code is within short tags, but will only work if short_open_tag is enabled
```

UNDERSTANDING FUNCTIONS

- A **function** is a subroutine (or individual statements grouped into a logical unit) that performs a specific task
  - To execute a function, you must invoke, or **call**, it from somewhere in the script
- A **function call** is the function name followed by any data that the function needs
- The data (in parentheses following the function name) are called **arguments** or **actual parameters**
- Sending data to a called function is called **passing arguments**

DISPLAYING SCRIPT RESULTS

- The `echo` and `print` statements are **language constructs** (built-in features of a programming language) that create new text on a Web page that is returned as a response to a client
- The text passed to the `echo` statement is called a "literal string" and must be enclosed in either single or double quotation marks
- To pass multiple arguments to the `echo` statement, separate the statements with commas

DISPLAYING SCRIPT RESULTS (CONTINUED)

- Use the `echo` and `print` statements to return the results of a PHP script within a Web page that is returned to a client
- The `print` statement returns a value of `1` if successful or a value of `0` if not successful, while the `echo` statement does not return a value

CREATING MULTIPLE CODE DECLARATION BLOCKS

- For multiple script sections in a document, include a separate code declaration block for each section

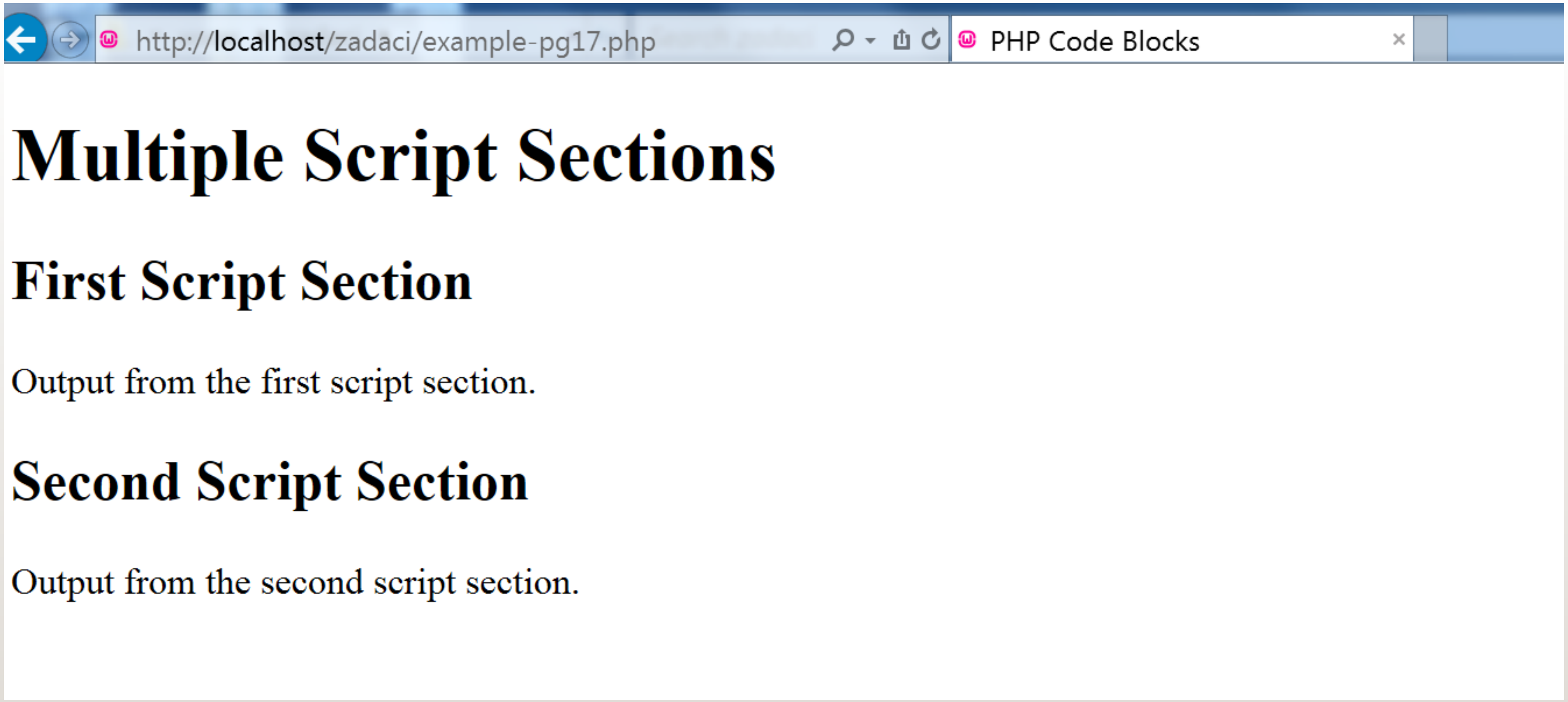
```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<?php
  echo "<p>Output from the first script section.</p>";
?>
<h2>Second Script Section</h2>
<?php
  echo "<p>Output from the second script section.</p>";
?>
</body>
</html>
```

CREATING MULTIPLE CODE DECLARATION BLOCKS

- PHP code declaration blocks execute on a Web server before a Web page is sent to a client

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<p>Output from the first script section.</p>
<h2>Second Script Section</h2>
<p>Output from the second script section.</p>
</body>
</html>
```

CREATING MULTIPLE CODE DECLARATION



Multiple Script Sections

First Script Section

Output from the first script section.

Second Script Section

Output from the second script section.

PHP BUILD-IN FUNCTIONS

- phpversion() - returns the version of PHP that processed the current page
- zend_version() - returns the version number of the Zend Engine (PHP's scripting engine)
- ini_get() function - returns the value assigned to a directive in the php.ini configuration file
    - You need to pass the name of a directive to the ini_get() function surrounded by quotation marks

CASE SENSITIVITY IN PHP

- Programming language constructs in PHP are mostly case **insensitive**

```
<?php
echo "<p>Explore <strong>Africa</strong>, <br />";
Echo "<strong>South America</strong>, <br />";
ECHO " and <strong>Australia</strong>!</p>";
?>
```

- Variables and constant name are case sensitive

ADDING COMMENTS TO A PHP SCRIPT

- **Comments** are nonprinting lines placed in code that do not get executed, but provide helpful information, such as:
  - The name of the script
  - Your name and the date you created the program
  - Notes to yourself
  - Instructions to future programmers who might need to modify your work
- **Line comments** hide a single line of code
  - Add // or # before the text
- **Block comments** hide multiple lines of code
  - Add /* to the first line of code
  - And */ after the last character in the code

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**
- The following rules and conventions must be followed when naming a variable:
  - Identifiers must begin with a dollar sign ($)
  - Identifiers may contain uppercase and lowercase letters, numbers, or underscores (_) - The first character after the dollar sign must be a letter
  - Identifiers cannot contain spaces
  - Identifiers are case sensitive

DECLARING AND INITIALIZING VARIABLES

- Specifying and creating a variable name is called **declaring the variable**
- Assigning a first value to a variable is called **initializing the variable**
- In PHP, you must declare and initialize a variable in the same statement:

```
$variable_name = value;
```

DISPLAYING VARIABLES

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

```
$VotingAge = 18;
echo $VotingAge;
```

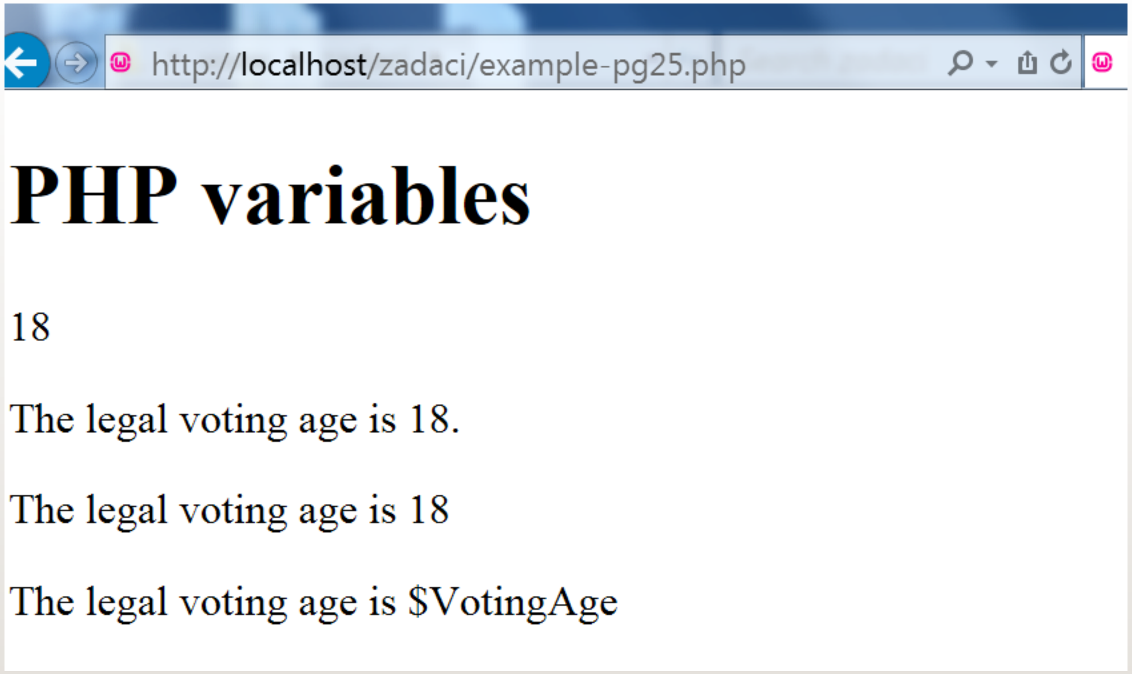
- To display both text strings and variables, send them to the `echo` statement as individual arguments, separated by commas

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

- Or include variable name inside a text string:

```
echo "<p>The legal voting age is $VotingAge</p>";
echo '<p>The legal voting age is $VotingAge</p>';
```

DISPLAYING VARIABLES



```
PHP variables

18

The legal voting age is 18.

The legal voting age is 18

The legal voting age is $VotingAge
```

MODIFYING VARIABLES

- You can modify a variable's value at any point in a script

```
$SalesTotal = 40;

echo "<p>Your sales total is
     $$SalesTotal</p>";

$SalesTotal = 50;

echo "<p>Your new sales total is
$$SalesTotal</p>";
```

DEFINING CONSTANTS

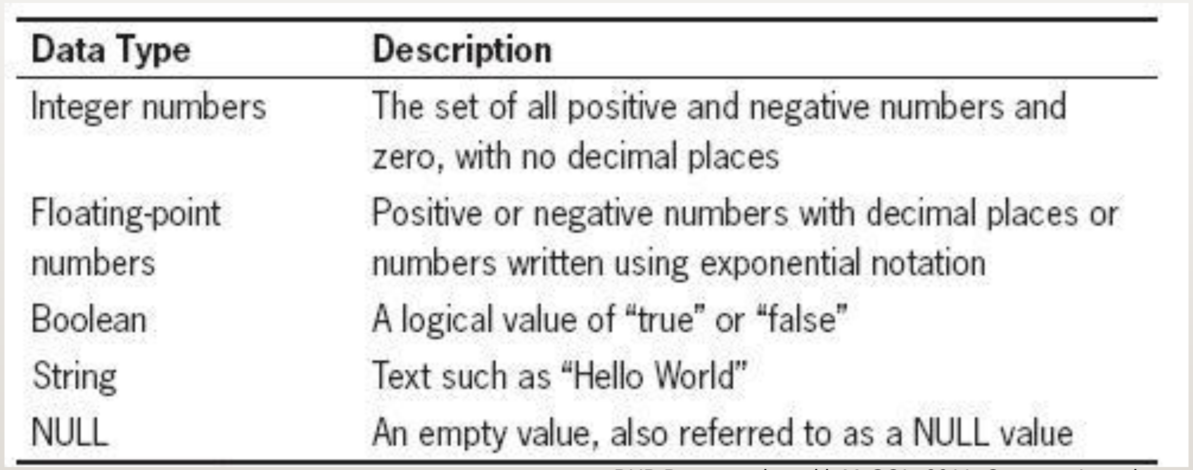
- A **constant** contains information that does not change during the course of program execution
- Constant names do not begin with a dollar sign ($)
- Constant names use all uppercase letters
- Use the **`define()`** function to create a constant

```
define("CONSTANT_NAME", value);
```

- The value you pass to the `define()` function can be a text string, number, or Boolean value
- Unlike variables, constant names cannot be included within the quotation marks of the `echo` statement

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**



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

PHP Programming with MySQL, 2011, Cengage Learning.

WORKING WITH DATA TYPES

- The PHP language supports:
  - **Reference** or **composite** data types, which contain multiple values or complex types of information
  - Two reference data types: **arrays** and **objects**
  - **“resource”** data type is a special variable that holds a reference to an external resource (e.g. XML file)

WORKING WITH DATA TYPES

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

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)

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` Boolean values

DECLARING AND INITIALIZING INDEXED ARRAYS

- An **array** contains a set of data represented by a single variable name
- An **element** refers to each piece of data that is stored within an array
- *In PHP the values assigned to different elements with same array can be of different types
- An **index** is an element's numeric position within the array
  - By default, indexes begin with the number zero (0)
  - An element is referenced by enclosing its index in brackets at the end of the array name:

    ```
    $Provinces[1]
    ```

DECLARING AND INITIALIZING INDEXED ARRAYS

- 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"
     );
```

DECLARING AND INITIALIZING INDEXED ARRAYS

- 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";
```

ACCESSING ELEMENT INFORMATION

- There are `print_r()`,`var_dump()` or `var_export()` functions to display or return information about variables
- The `print_r()` function displays the index and value of each element in an array
- The `var_dump()` function displays the index, value, data type and number of characters in the value
- The `var_export()` function is similar to `var_dump()` function except returned representation is a valid PHP code

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";
```

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";
```

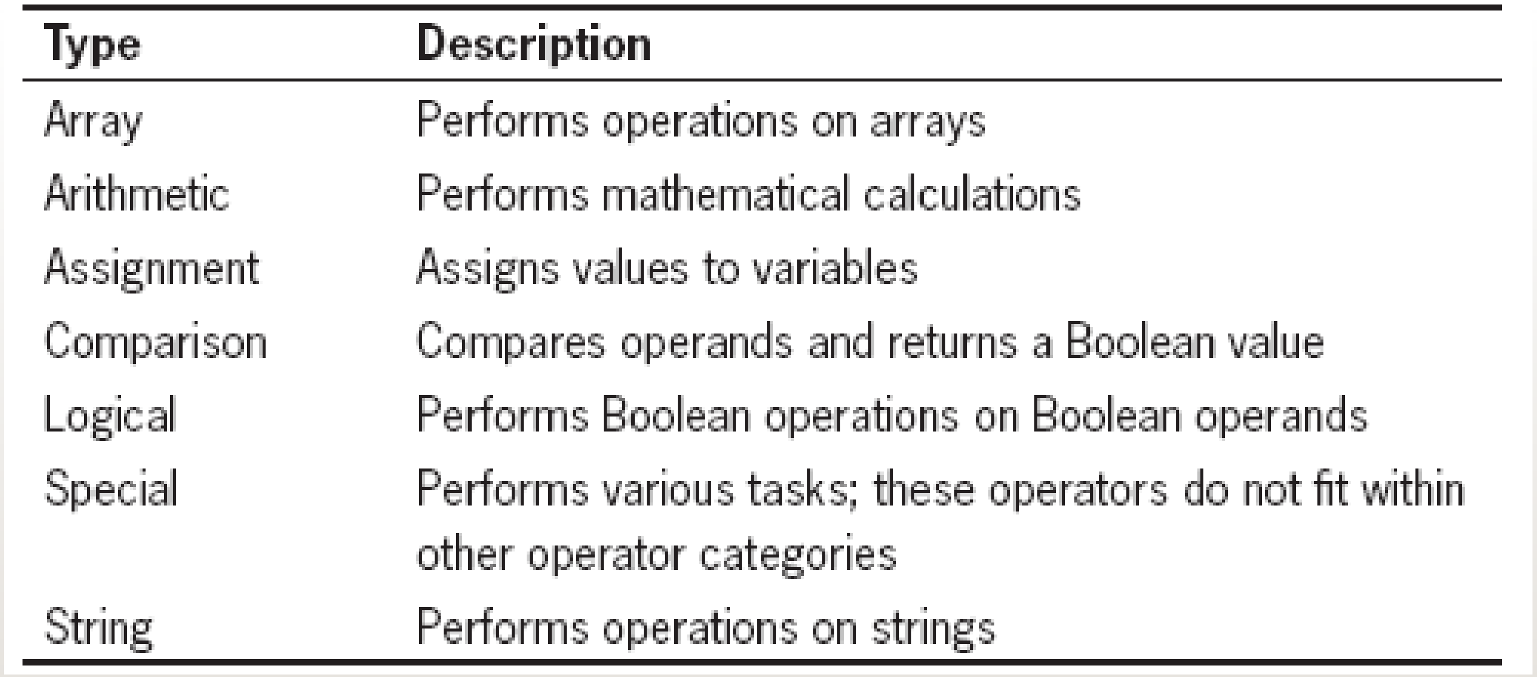
- Replaces the value stored in the first element (index 0) of the $list array with the string "Hello"

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

BUILDING EXPRESSIONS

- An **expression** is a literal value or variable (or a combination of literal values, variables, operators or other expressions) 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 OPERATOR TYPES



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

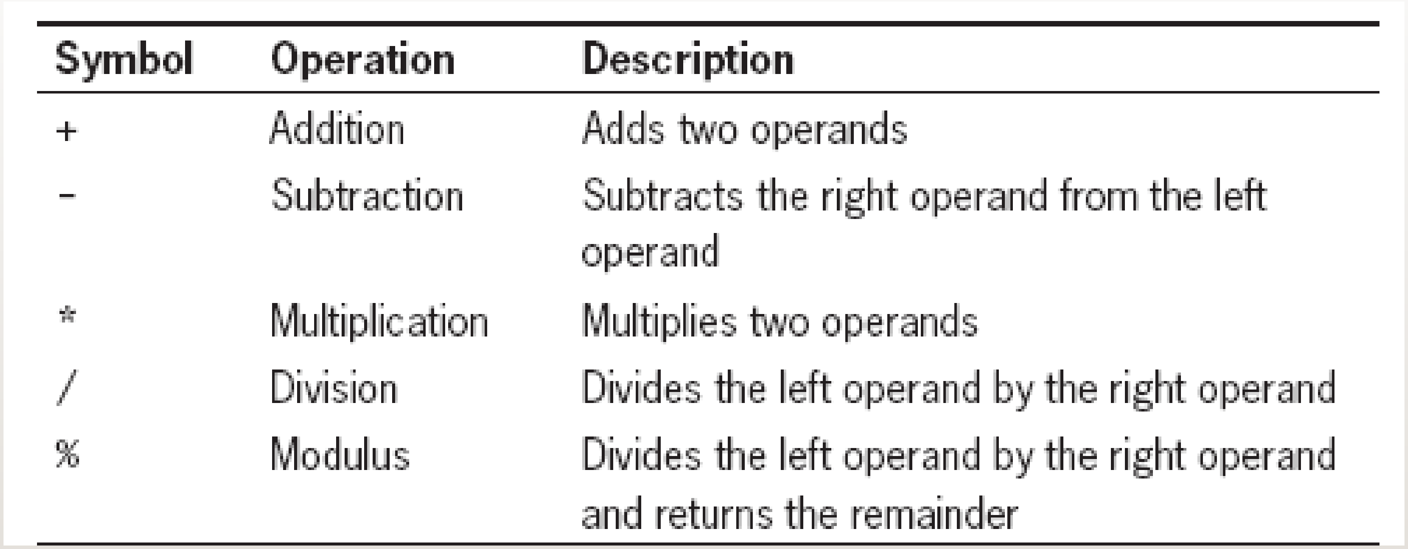
PHP Programming with MySQL, 2011, Cengage Learning.

BUILDING EXPRESSIONS

- 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

ARITHMETIC BINARY OPERATORS

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

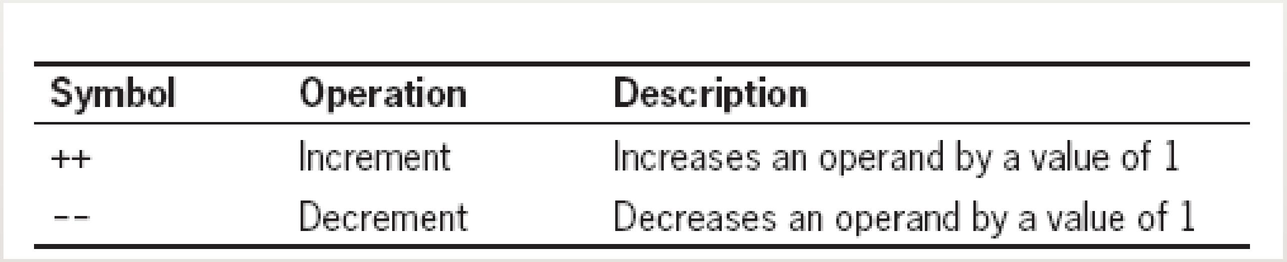


Symbol	Operation	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts the right operand from the left operand
*	Multiplication	Multiplies two operands
/	Division	Divides the left operand by the right operand
%	Modulus	Divides the left operand by the right operand and returns the remainder

PHP Programming with MySQL, 2011, Cengage Learning.

ARITHMETIC UNARY OPERATORS

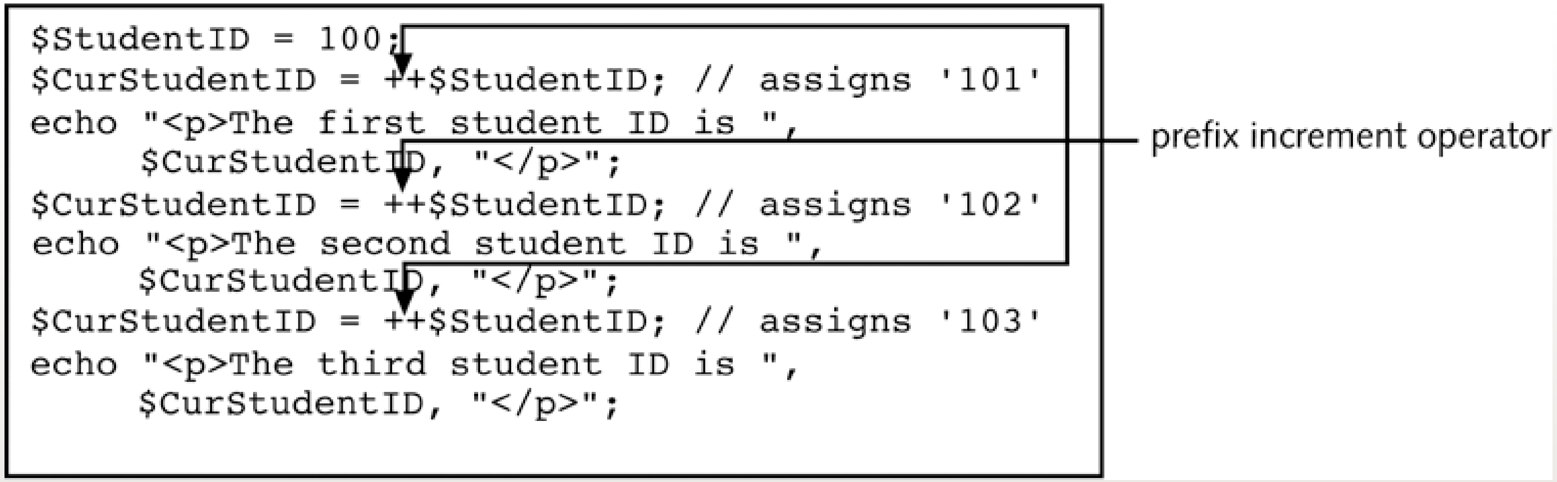
- The increment (++) and decrement (--) unary operators can be used as prefix or postfix operators
- A **prefix operator** is placed before a variable
- A **postfix operator** is placed after a variable



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

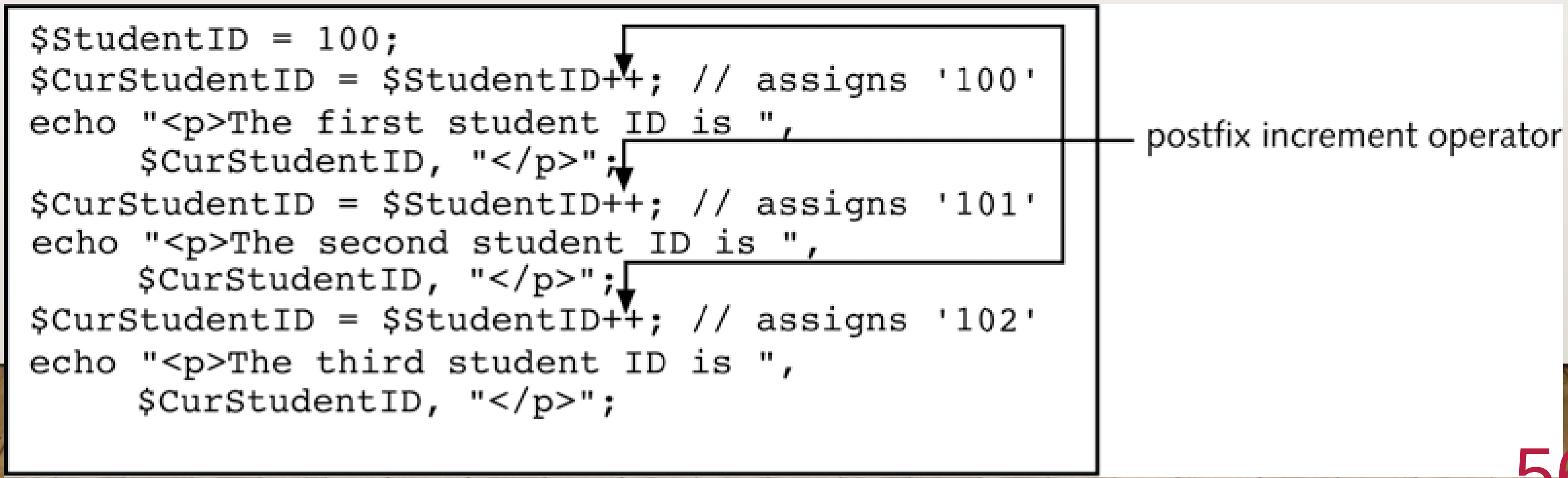
PHP Programming with MySQL, 2011, Cengage Learning.

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



```
$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>";
```

postfix increment operator

ASSIGNMENT OPERATORS

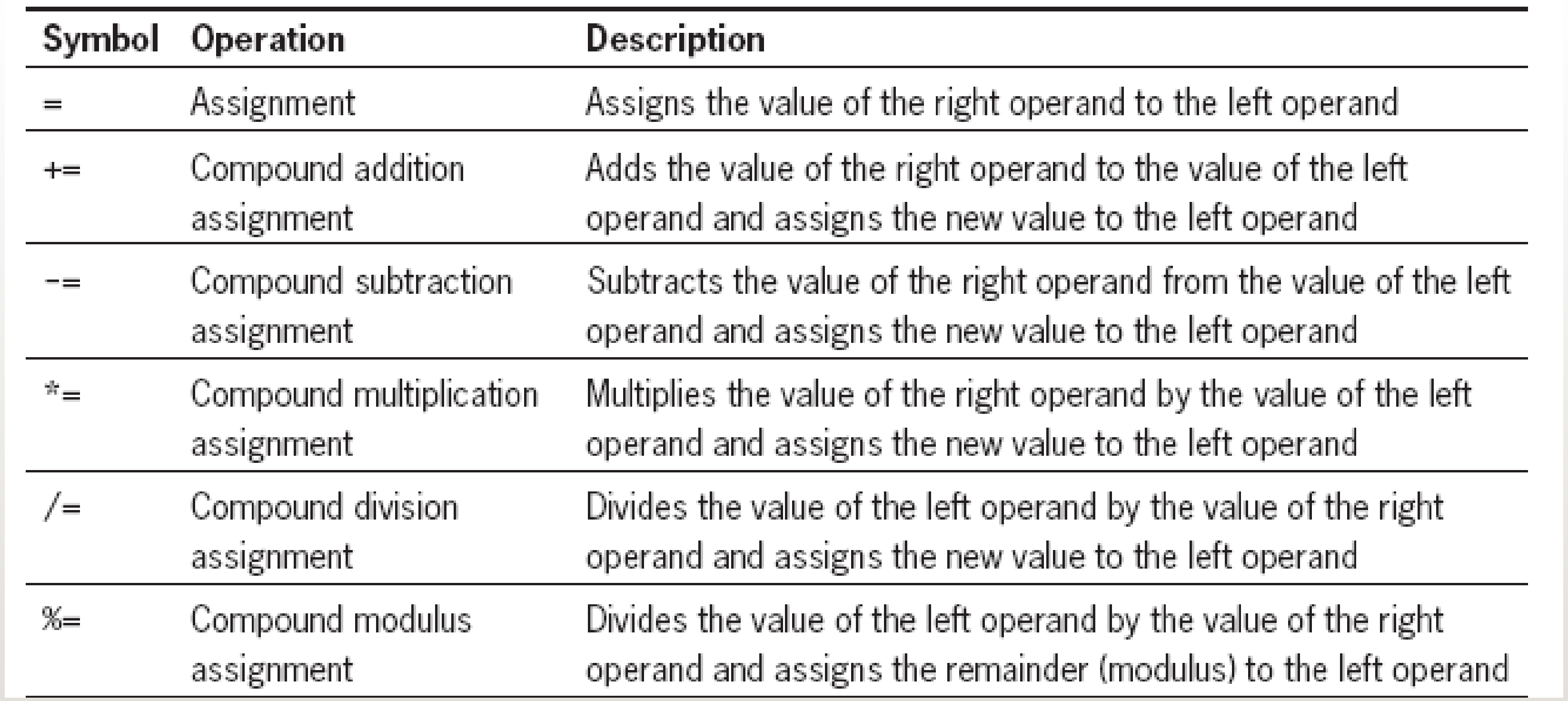
- **Assignment operators** are used for assigning a value to a variable:

```
$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

```
$a += $b;
```

ASSIGNMENT OPERATORS (CONTINUED)



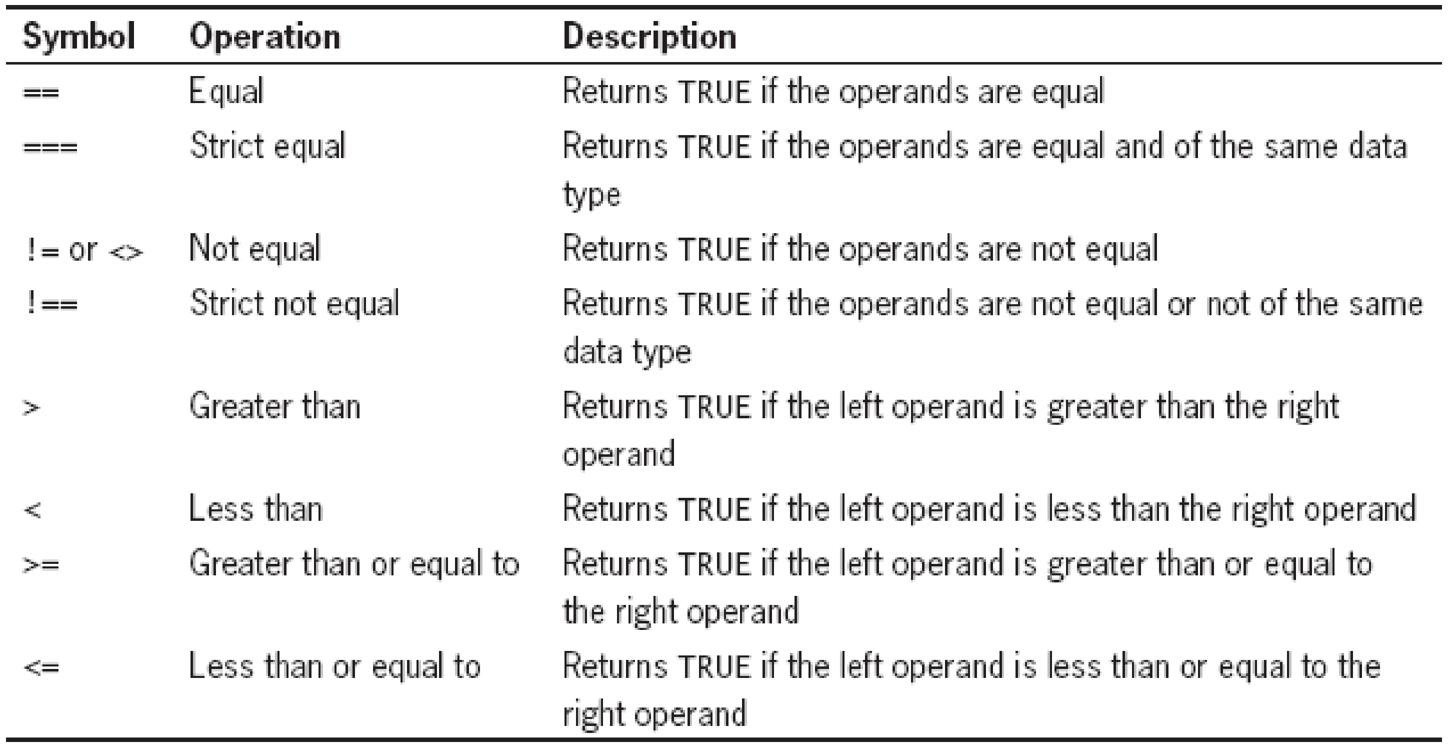
Symbol	Operation	Description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the new value to the left operand
-=	Compound subtraction assignment	Subtracts the value 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

PHP Programming with MySQL, 2011, Cengage Learning.

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



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

<=>	The **rocket ship operator** (available in PHP7+)

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;
```

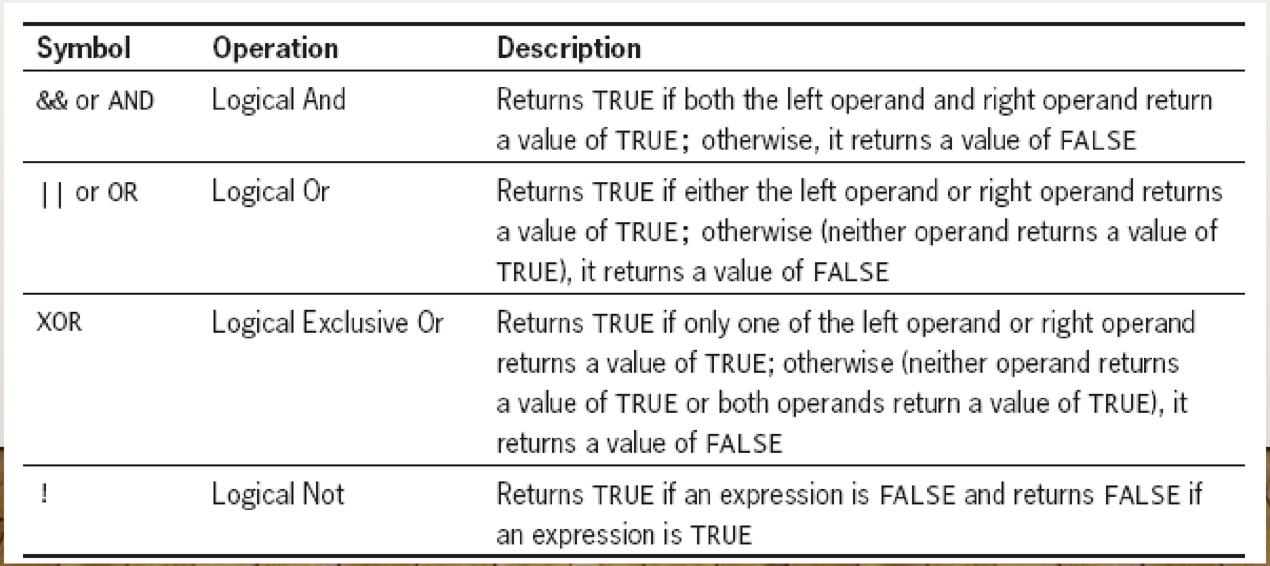
- If the conditional expression evaluates to `TRUE`, *`expression1`* executes
- If the conditional expression evaluates to `FALSE`, *`expression2`* executes

COMPARISON AND CONDITIONAL OPERATORS

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

LOGICAL OPERATORS

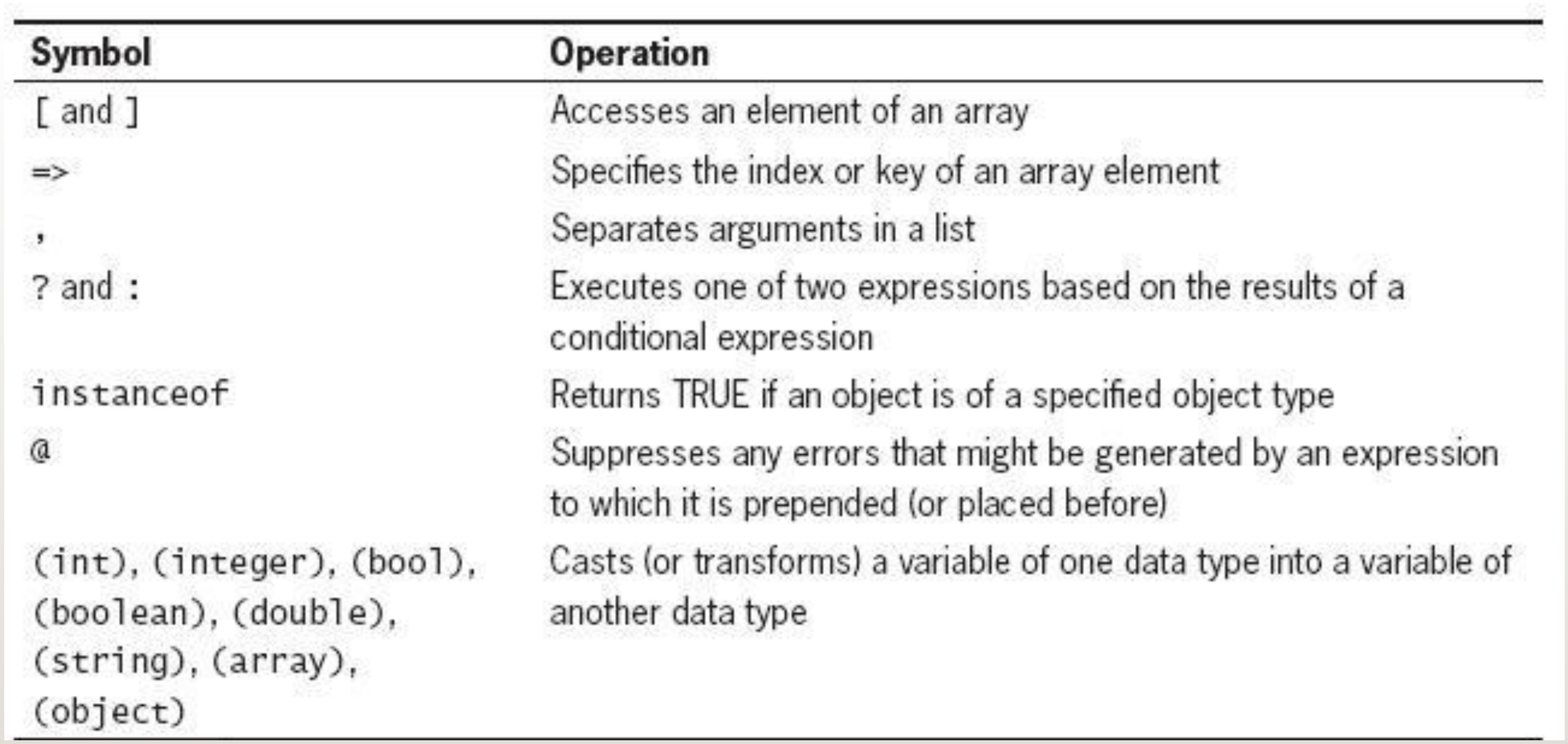
- **Logical operators** are used for comparing two Boolean operands for equality
- A Boolean value of `TRUE` or `FALSE` is returned after two operands are compared



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

PHP Programming with MySQL, 2011, Cengage Learning.

SPECIAL OPERATORS



Symbol	Operation
[and]	Accesses an element of an array
=>	Specifies the index or key of an array element
,	Separates arguments in a list
? and :	Executes one of two expressions based on the results of a conditional expression
`instanceof`	Returns TRUE if an object is of a specified object type
@	Suppresses 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

PHP Programming with MySQL, 2011, Cengage Learning.

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
- PHP can convert string into numeric value if the string starts with numeric value, any subsequent non-numeric characters are ignored

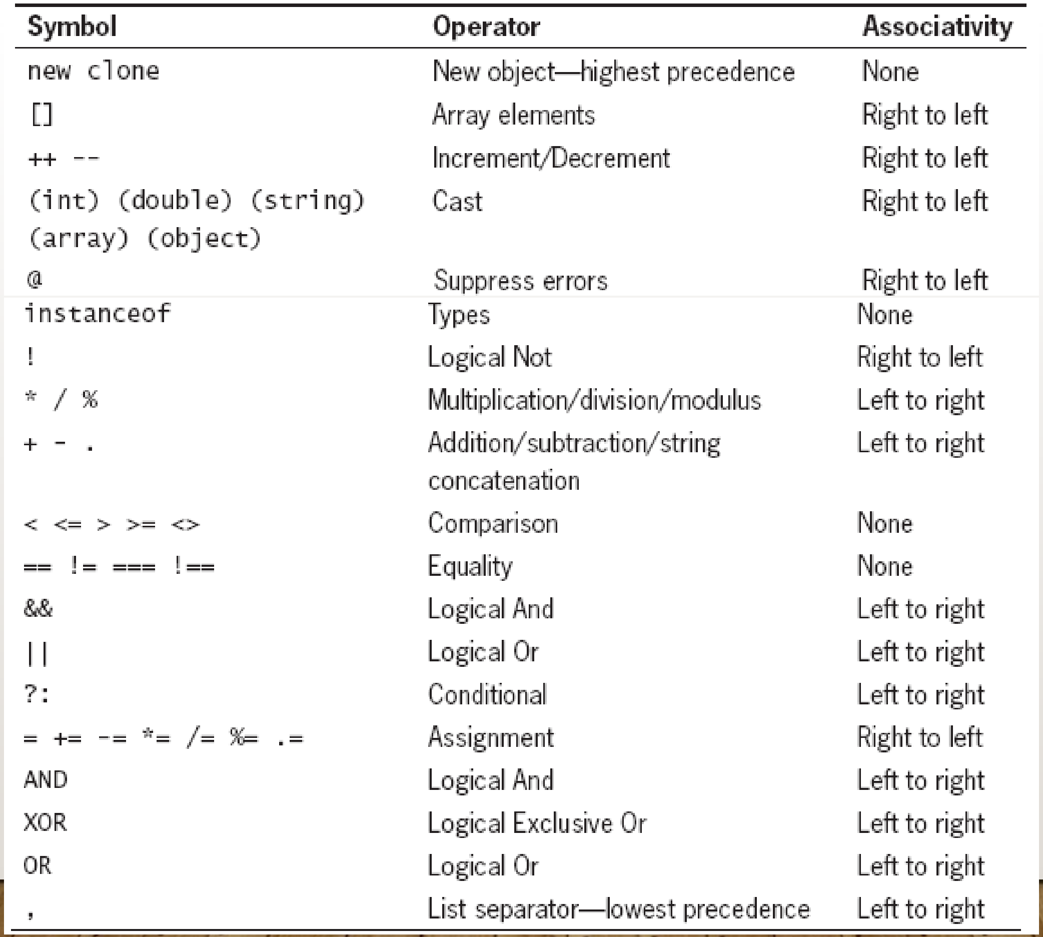
TYPE CASTING – `GETTYPE()` FUNCTION

- Returns one of the following strings, depending on the data type:
  - Boolean
  - Integer
  - Double
  - String
  - Array
  - Object
  - Resource
  - NULL
  - Unknown type
- Also can be used `is_*()` function
  - `is_numeric($a), is_int($a), is_string($a)`

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)



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
`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

PHP Programming with MySQL, 2011, Cengage Learning.