MIS6070

Web Based Information Systems

Lesson 7

JavaScript and Client-Side Scripting

- **JavaScript** is:
  - A client-side scripting language that allows Web page authors to develop interactive Web pages and sites
  - Used in most Web browsers including Firefox and Internet Explorer
- **Client-side scripting** is a language that runs on a local browser (on the client tier) instead of on a Web server (on the processing tier)

JavaScript and Client-Side Scripting

- JavaScript allows you to:
  - Turn static Web pages into applications such as games or calculators
  - Change the contents of a Web page after a browser has rendered it
  - Create visual effects such as animation
  - Control the Web browser window itself

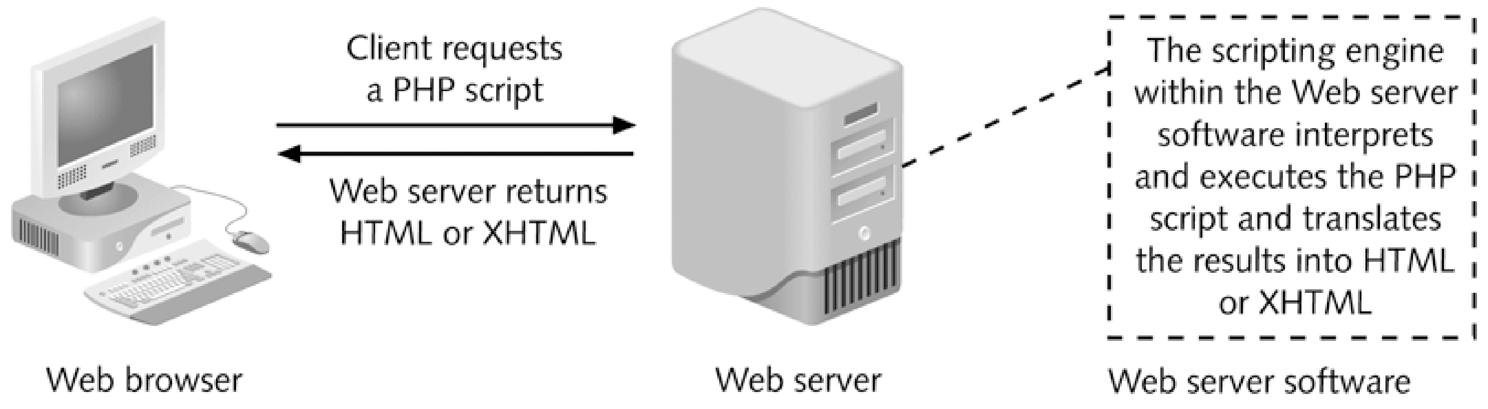
Server-Side Scripting and PHP

- **Server-side scripting** refers to a scripting language that is executed from a Web server
- **Hypertext Preprocessor (PHP)** is a server-side scripting language that is used to develop interactive Web sites
  - Is easy to learn
  - Includes object-oriented programming capabilities
  - Supports many types of databases (MySQL, Oracle, Sybase, ODBC-compliant)

Server-Side Scripting and PHP

- PHP (continued):
  - PHP is an **open source** programming language
    - Open source refers to software where source code can be freely used and modified
  - Can’t access or manipulate a Web browser like JavaScript
  - Exists and executes solely on a Web server, where it performs various types of processing or accesses databases

Server-Side Scripting and PHP


Client requests a PHP script
Web server returns HTML or XHTML
The scripting engine within the Web server software interprets and executes the PHP script and translates the results into HTML or XHTML
Web browser
Web server
Web server software

How a Web server processes a PHP script

- **General rule:** Use client-side scripting to handle user interface processing and light processing, such as validation; use server-side scripting for intensive calculations and data storage

Creating Basic PHP Scripts

- PHP
  - PHP: Hypertext Preprocessor
  - Originally called “Personal Home Page Tools”
  - Popular **server-side scripting** technology
  - Open-source
    - Anyone may view, modify and redistribute source code
    - Supported freely by community
  - Platform independent

Creating Basic PHP Scripts

- **An introductory example**

```
<html>
  <head>
   <title>PHP Example</title>
  </head>
  <body>
   <?php
          echo "Hi, I’m a PHP script!";
   ?>
  </body>
</html>
```

- The semicolon (;) signifies the end of a PHP statement and should never be forgotten.

Creating Basic PHP Scripts

- Can be embedded into HTML.
  - Write an HTML script with some embedded code to do something
  - Code is executed on the server.
- **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 code is never sent to a client's Web browser; **only the output of the processing is sent to the browser**

Creating Basic PHP Scripts

- The Web page generated from the PHP code, and HTML elements found within the PHP file, is returned to the client
- A PHP 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
  - The `<script>` element
  - Short PHP script delimiters
  - ASP-style script delimiters

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**

The `<script>` Element

- The **`<script>` element** identifies a script section in a Web page document
- Assign a value of "php" to the **language** attribute of the `<script>` element to identify the code block as PHP
- `<script>`
  - `//<some valid PHP syntax>`
- `</script>`

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 inHTML, XHTML documents, but not in XML documents

ASP-Style Script Delimiters

- The syntax for the ASP-style script delimiters is

  `<% statements; %>`

- ASP-style script delimiters can be used in XHTML documents, but not in XML documents
- ASP-style script delimiters can be enabled or disabled in the php.ini configuration file
- To enable or disable ASP-style script delimiters, assign a value of “On” or “Off ” to the `asp_tags` directive in the php.ini configuration file

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 commas like arguments passed to a function

Displaying Script Results

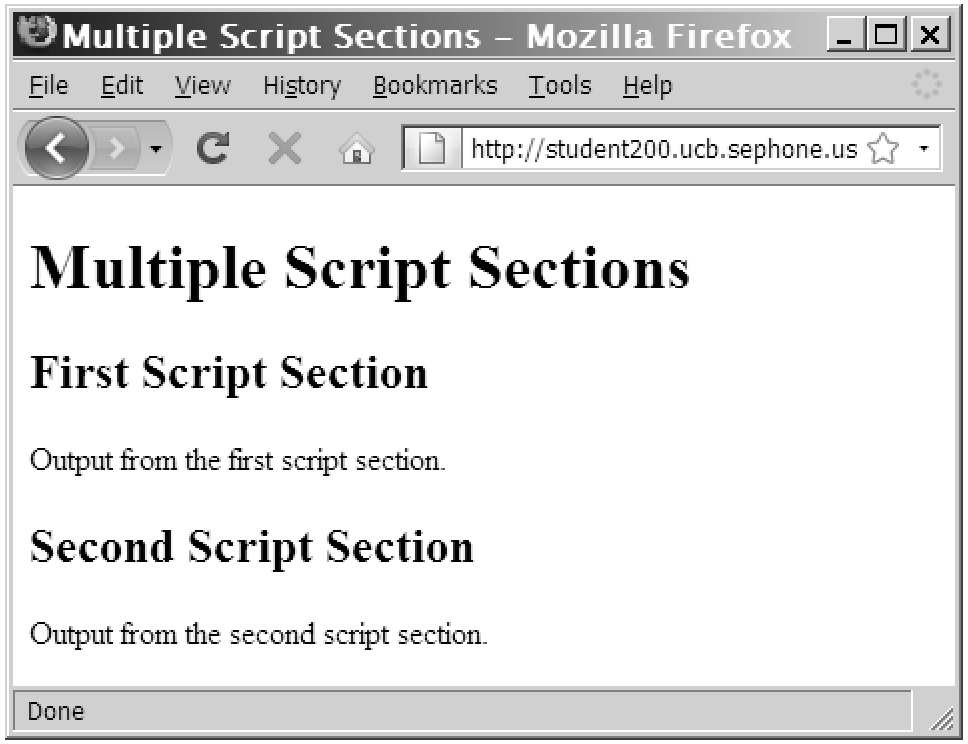
- Use the `echo` and `print` statements to return the results of a PHP script within a Web page that is returned to a client
- The `print` statement returns a value of `1` if successful or a value of `0` if not successful, while the `echo` statement does not return a value
- Echo ‘Hello , World’;
- Print ‘Hello, World’;
    - Either ‘ or “

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 Blocks


Multiple Script Sections – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.us
Done

```
Multiple Script Sections

First Script Section

Output from the first script section.

Second Script Section

Output from the second script section.
```

Output of a document with two PHP script sections

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>
```

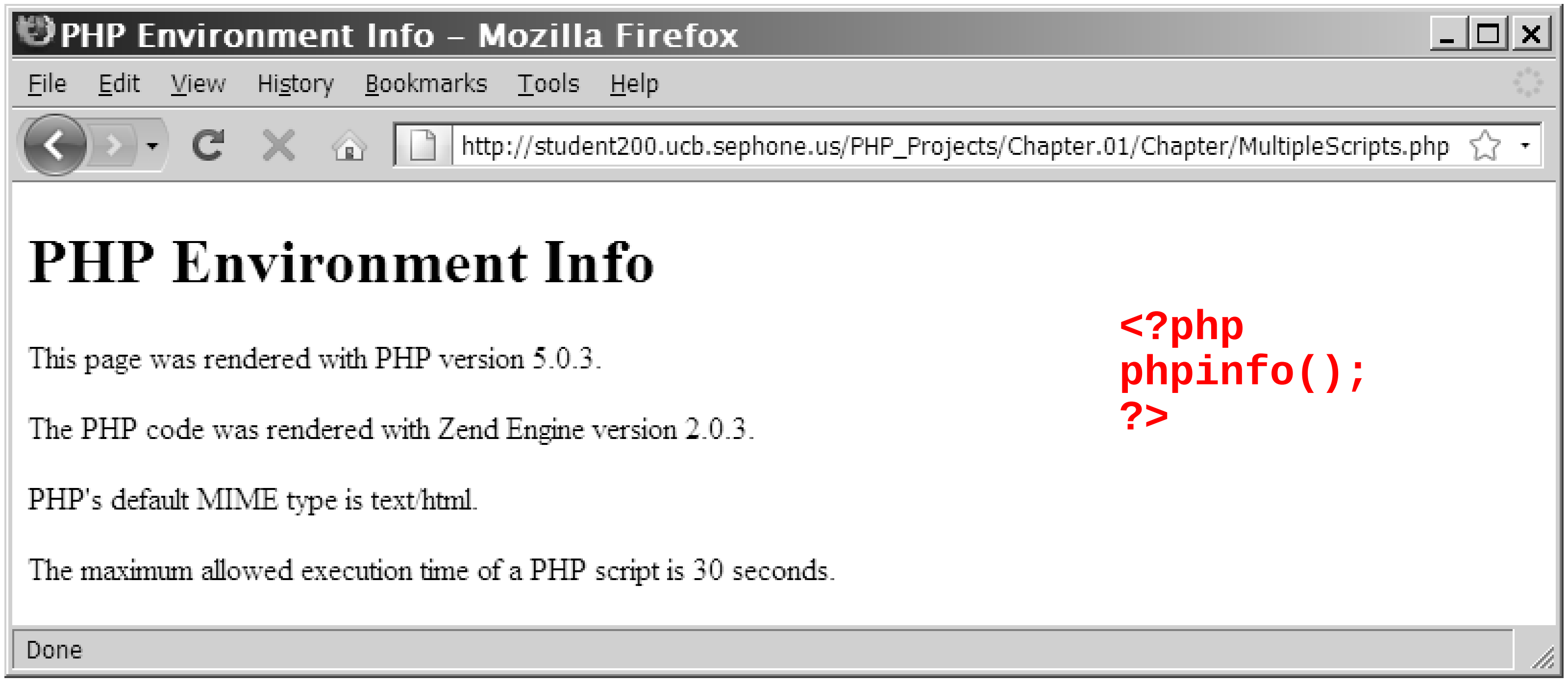
Displaying Variables

- You can use Echo or Print to print a value of the variable.

```
<?php
echo ‘Hello World!<br />’;
echo(‘Hello World!<br />’);
print ‘Hello World!<br />’;
print(‘Hello World!<br />’);
?>
```

- The difference is that the echo statement can take multiple arguments separated by commas whereas the print statement cannot take multiple value arguments
- Example
  - Echo $x, $y;//Valid statement
  - Print $x, $y;//Invalid statement
  - Print $x;//Valid statement

Creating Multiple Code Declaration Blocks


PHP Environment Info – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.us/PHP_Projects/Chapter.01/Chapter/MultipleScripts.php
Done

PHP Environment Info

This page was rendered with PHP version 5.0.3.

The PHP code was rendered with Zend Engine version 2.0.3.

PHP's default MIME type is text/html.

The maximum allowed execution time of a PHP script is 30 seconds.

```
<?php
phpinfo();
?>
```

PHP Environment Information Web page

Case Sensitivity in PHP

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

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

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

Adding Comments to a PHP Script

- **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

Adding Comments to a PHP Script

```
<?php
/*
This line is part of the block comment.
This line is also part of the block comment.
*/
echo "<h1>Comments Example</h1>"; // Line comments can
  follow
code statements
// This line comment takes up an entire line.
# This is another way of creating a line comment.
/* This is another way of creating
a block comment. */
?>
```

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**
- An identifier must begin with a dollar sign ($), may not include a number or underscore as the first character, cannot include spaces, and is case sensitive

Displaying Variables

- To display 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 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>";
```

Example

```
<?php
$name = 'PHP DEMO'; // assigning value to variable
echo $name; //calling the variable
?>
```

Naming Variables

- 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;
```

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>";
```

- You can destroy a variable
  - You destroy a variable using the unset() function

```
<?php
$name = 'we are USIU-Africa APT1040 Class'; // assign value to variable
echo "Before destroying, $name"."</br>";// print variable value
unset($name); // destroy variable
echo "After destroying, $name";// print variable value
?>
```

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

Defining Constants

```
<?php
define('NAME','Amandeep');
define('AGE',23);
echo NAME;
echo ' is ';
echo AGE;
// Amandeep is 23
?>
```

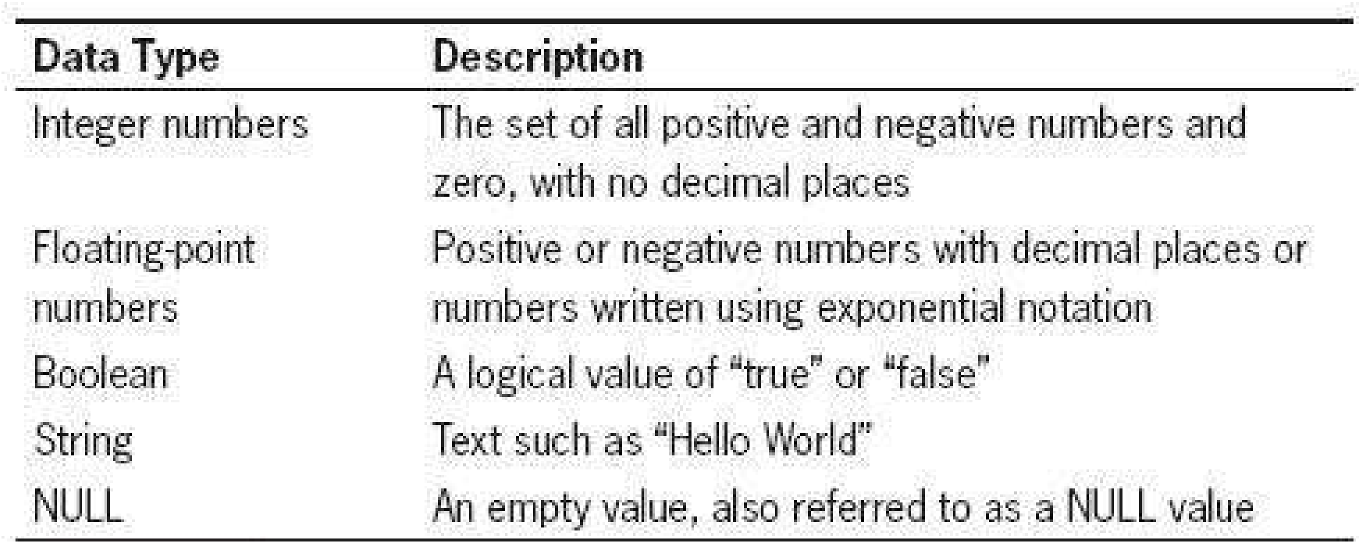
Working with Data Types

- There is a difference between strings written in single and double quotes.
- In a double-quoted string any variable names are expanded to their values.
- In a single-quoted string, no variable expansion takes place.

```
<?php
$name = ‘Amandeep’;
$age  = 23;
echo “$name is $age”;
// Amandeep is 23
echo ‘$name is $age’;
// $name is $age
?>
```

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

Working with Data Types

- 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**
    - **objects**

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 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 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
- In other programming languages, you can use integers such as 1 = TRUE, 0 = FALSE

Using type casting

- PHP allows you to convert the data type of a variable to another data type

```
<?php
$x = "3";  // $x is string (ASCII 48)
echo "$x, x is a string"."</br>";
$x += 3;   // $x is now an integer (3)
echo "$x, x is now an integer"."</br>";
$x = $x + 1.3;  // $x is now a float (3.3)
echo "$x, x is now a float";
?>
```

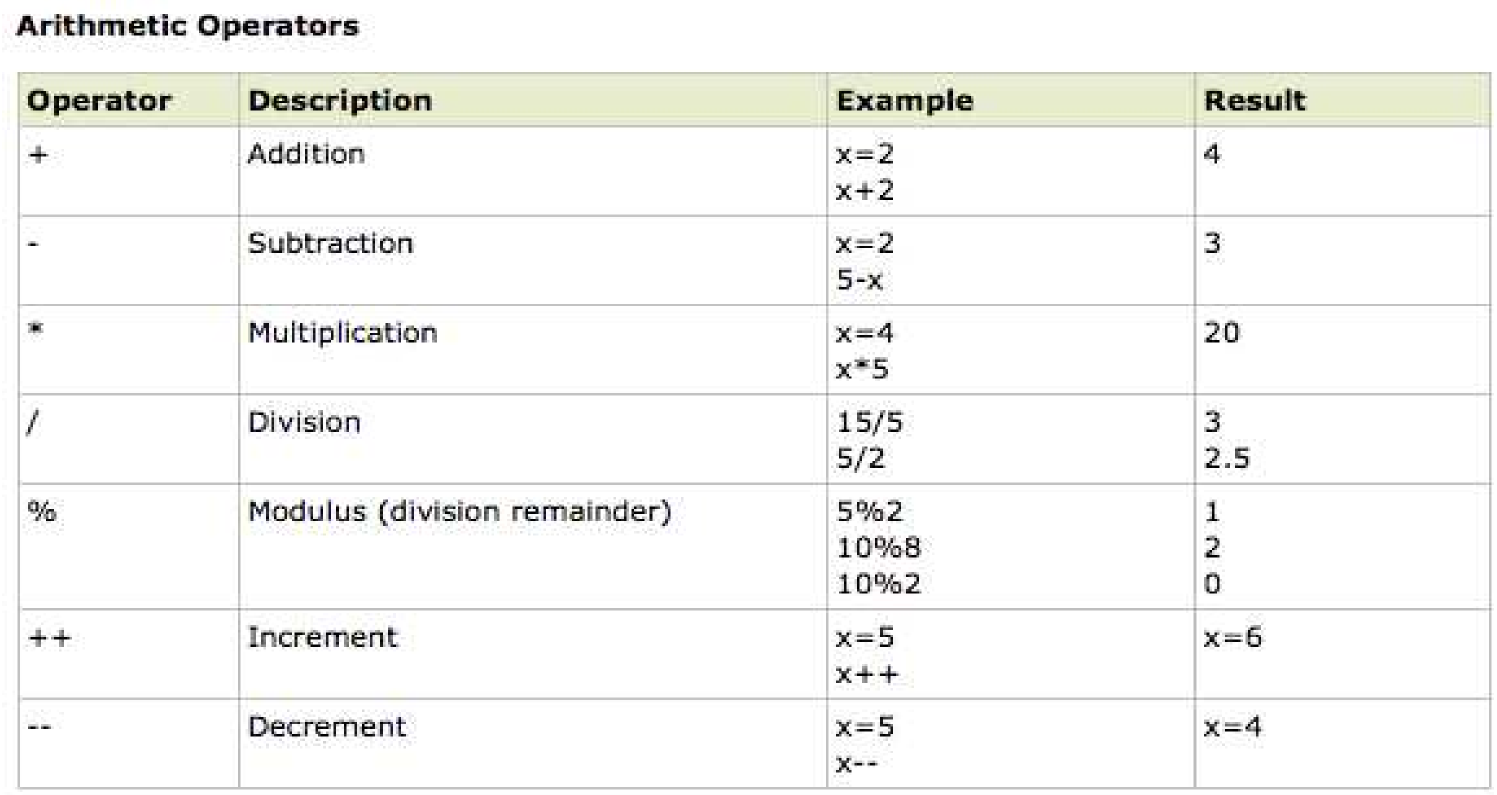
PHP Concatenation

- The concatenation operator (.) is used to put two string values together.
- To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello APT1040";
$txt2="What a nice day!";
echo $txt1 .  " " . $txt2;
?>
```

- If we look at the code you see that we used the concatenation operator two times.
  - This is because we had to insert a third string (a space character), to separate the two strings.

PHP Operators



Arithmetic Operators

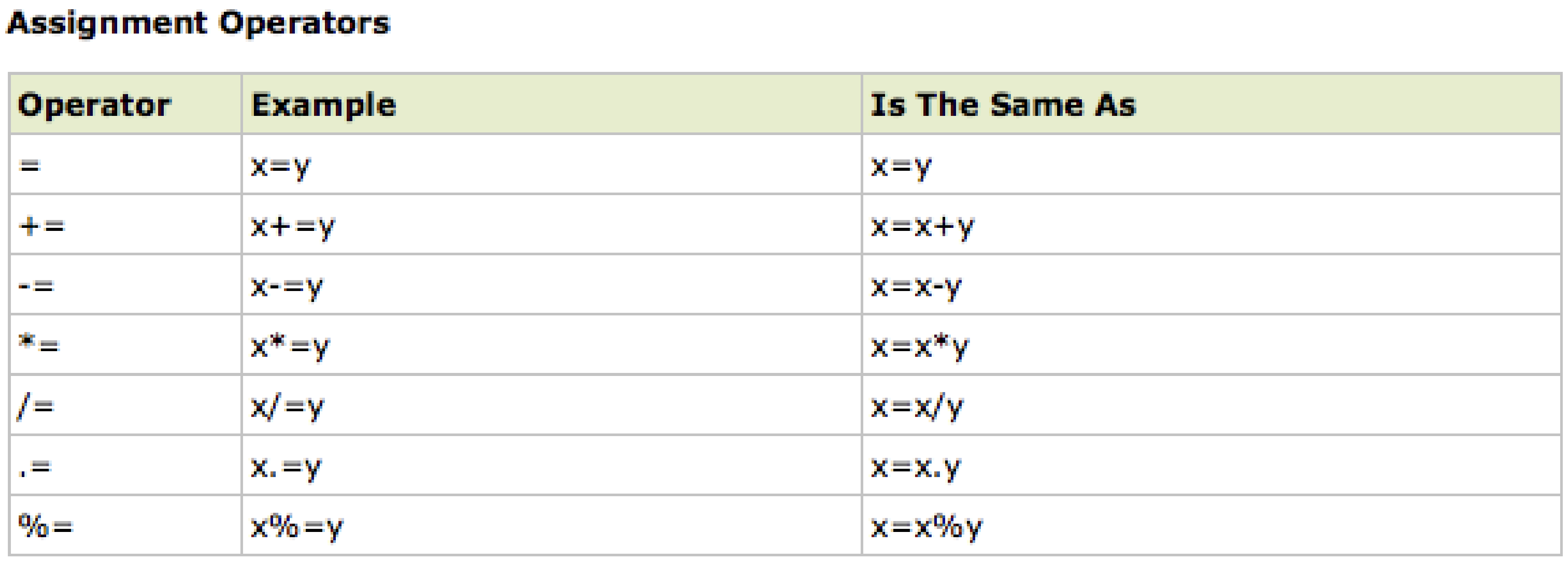
Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

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

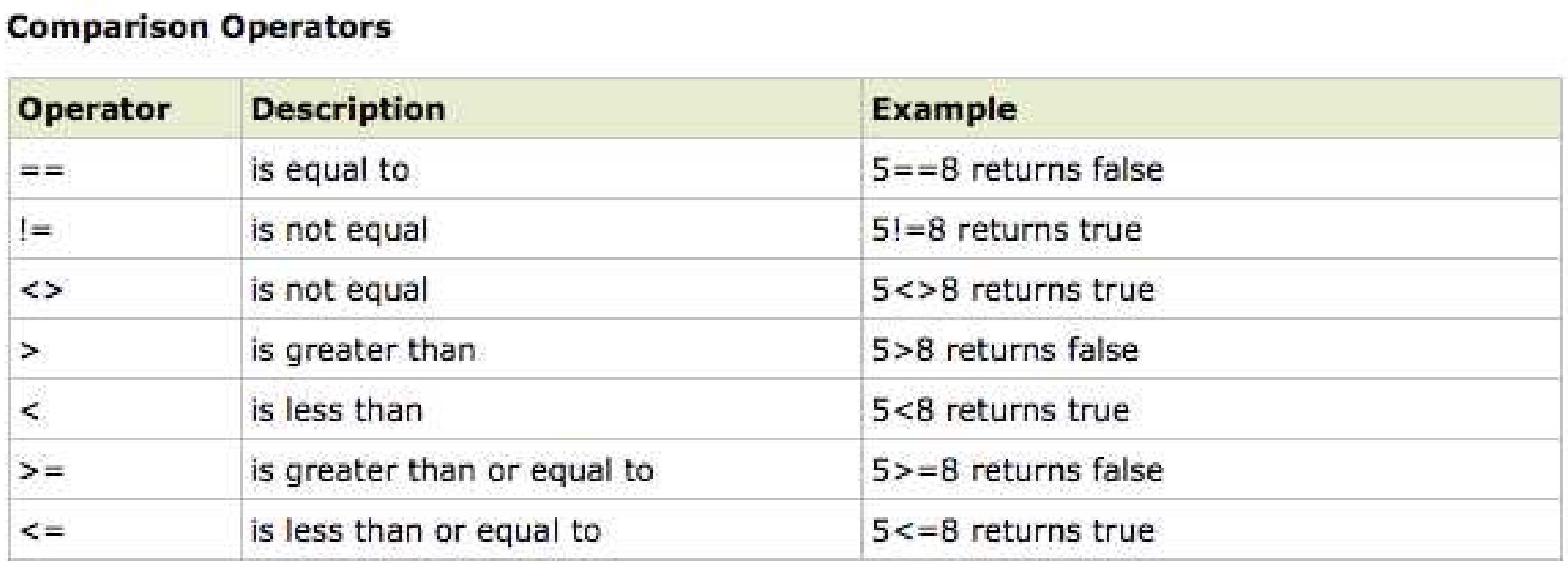
PHP Operators



Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

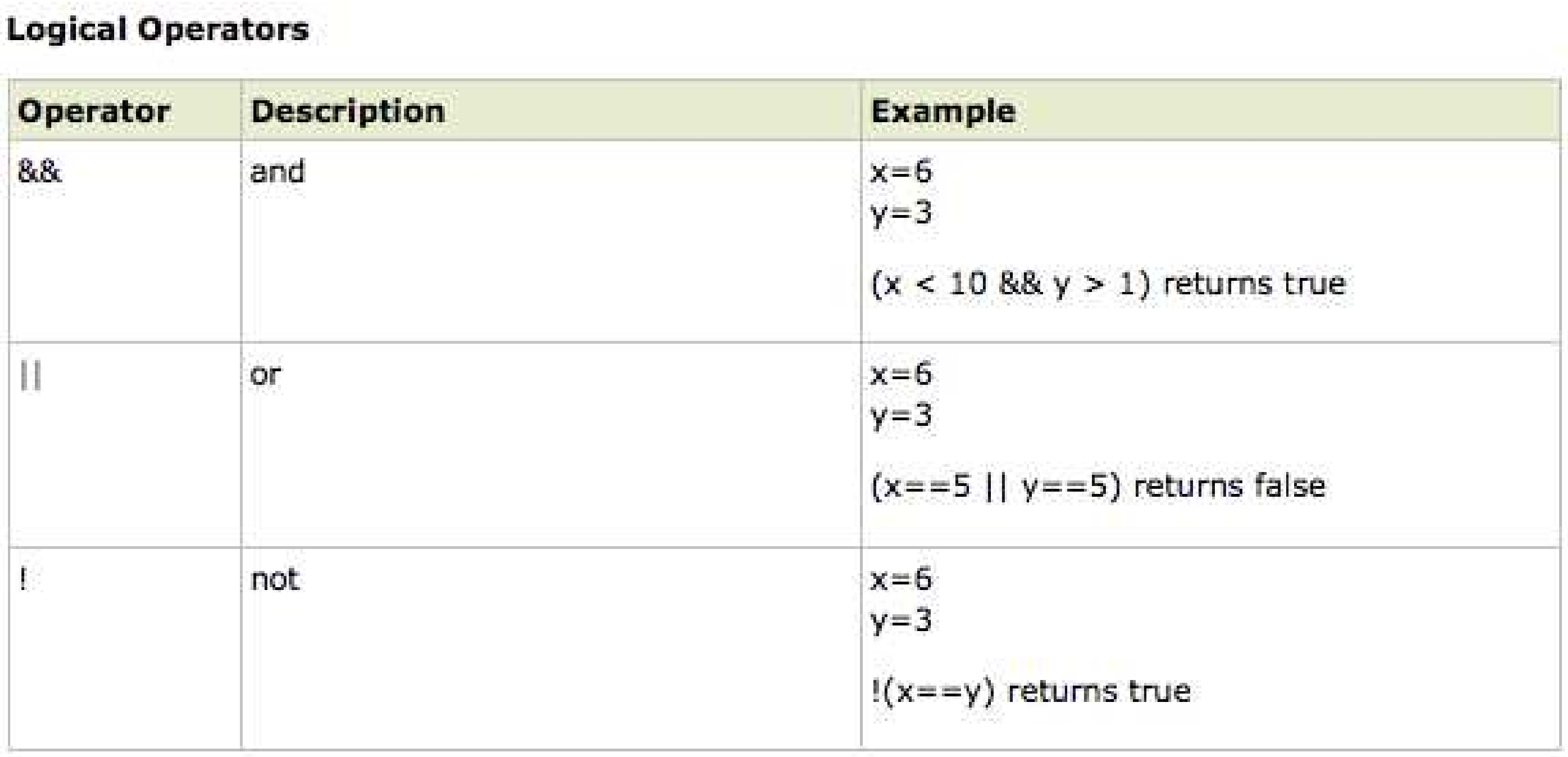
PHP Operators



Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

PHP Operators



Logical Operators

Operator	Description	Example

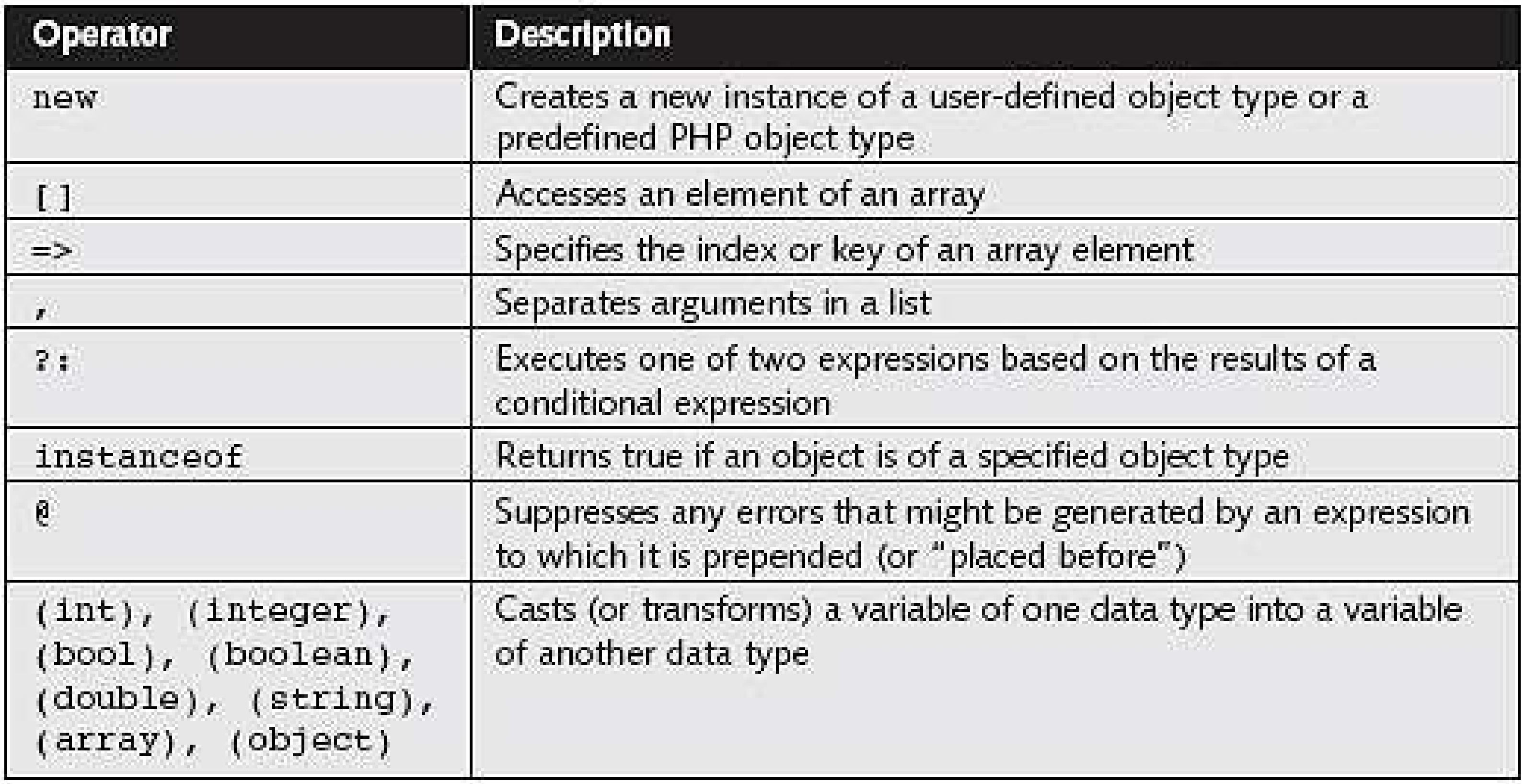
&&	and	x=6
		y=3
		(x < 10 && y > 1) returns true

||	or	x=6
		y=3
		(x==5 || y==5) returns false

!	not	x=6
		y=3
		!(x==y) returns true

PHP Operators

PHP special operators



Operator	Description
`new`	Creates a new instance 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 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

Defining Functions

- **Functions** are groups of statements that you can execute as a single unit
- **Function definitions** are the lines of code that make up a function
- The syntax for defining a function is:

```
<?php
function name_of_function(parameters) {
     statements;
}
?>
```

Defining Functions (continued)

- Functions, like all PHP code, must be contained within `<?php ... ?>` tags
- A **parameter** is a variable that is used within a function
- Parameters are placed within the parentheses that follow the function name
- Functions do not have to contain parameters
- The set of curly braces (called **function braces**) contain the function statements

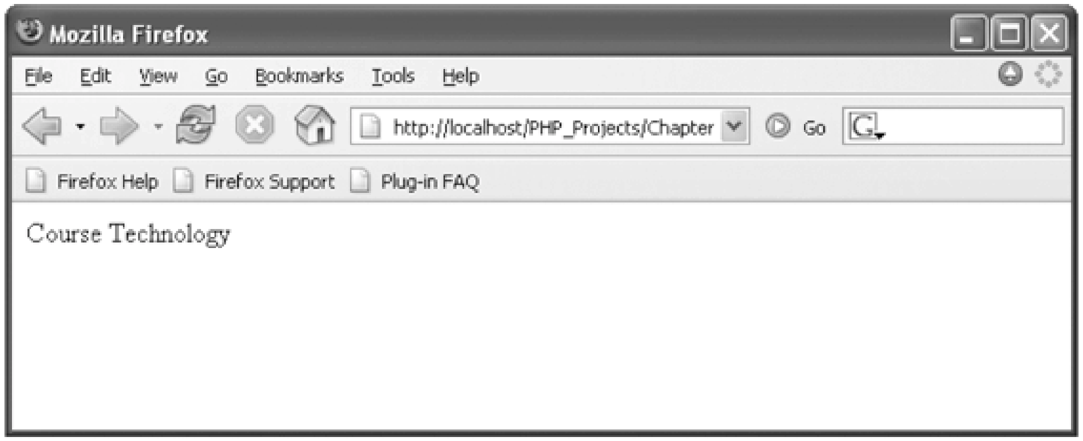
Defining Functions (continued)

- **Function statements** do the actual work of the function and must be contained within the function braces

```
function printCompanyName($Company1, $Company2, $Company3) {
     echo “<p>$Company1</p>”;
     echo “<p>$Company2</p>”;
     echo “<p>$Company3</p>”;
}
```

Calling Functions

```
function printCompanyName($CompanyName) {
      echo “<p>$CompanyName</p>”;
}
printCompanyName(“Course Technology”);
```


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

```
Course Technology
```

Output of a call to a custom function

Returning Values

- A **return statement** is a statement that returns a value to the statement that called the function
- A function does not necessarily have to return a value

```
   function averageNumbers($a, $b, $c) {
       $SumOfNumbers = $a + $b + $c;
       $Result = $SumOfNumbers / 3;
       Return $Result;
 }
```

Understanding Variable Scope

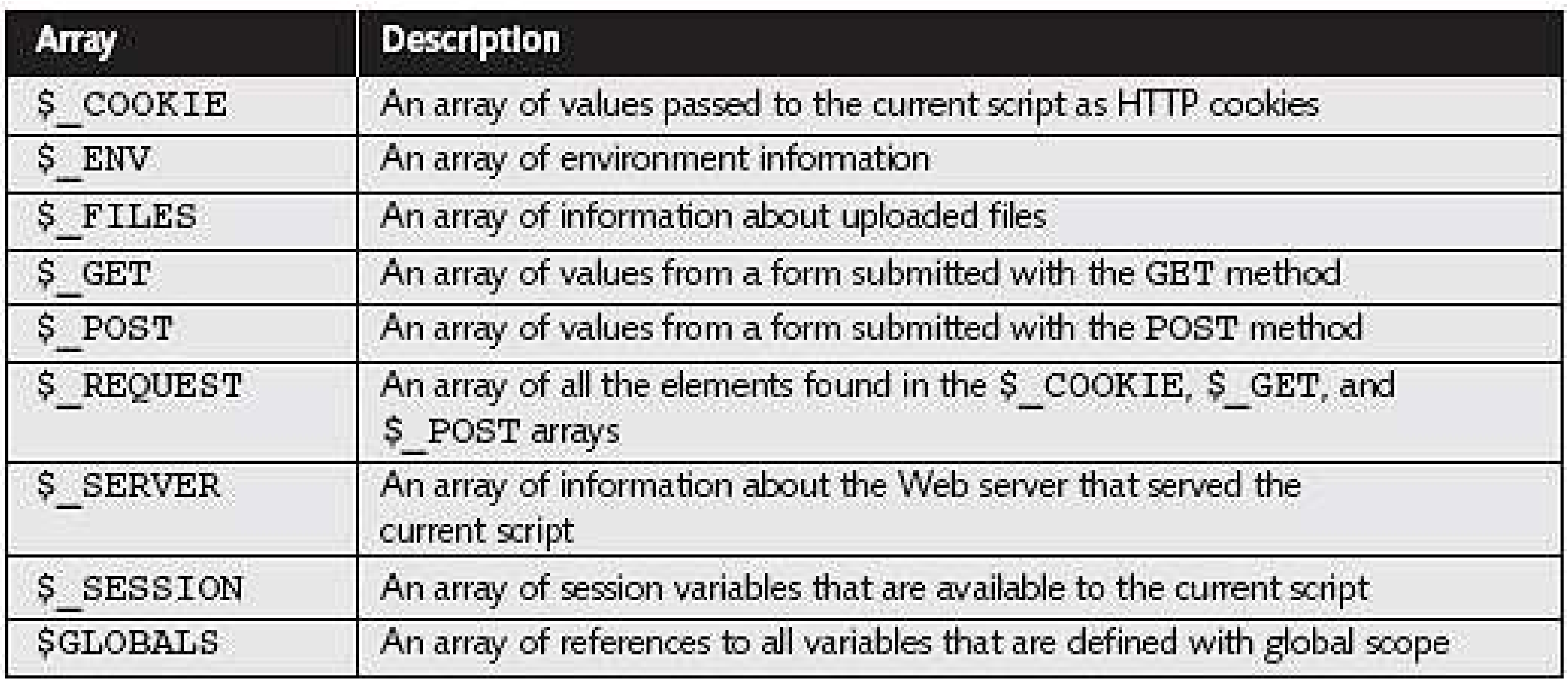
- **Variable scope** is where in your program a declared variable can be used
- A variable’s scope can be either global or local
- A **global variable** is one that is declared outside a function and is available to all parts of your program
- A **local variable** is declared inside a function and is only available within the function in which it is declared

Using Autoglobals

- PHP includes various predefined global arrays, called **autoglobals** or **superglobals**
- Autoglobals contain client, server, and environment information that you can use in your scripts
- Autoglobals are **associative arrays** – arrays whose elements are referred to with an alphanumeric key instead of an index number

Using Autoglobals (continued)

PHP autoglobals



Array	Description
`$_COOKIE`	An array of values passed to the current script as HTTP cookies
`$_ENV`	An array of environment information
`$_FILES`	An array of information about uploaded files
`$_GET`	An array of values from a form submitted with the `GET` method
`$_POST`	An array of values from a form submitted with the `POST` method
`$_REQUEST`	An array of all the elements found in the `$_COOKIE`, `$_GET`, and `$_POST` arrays
`$_SERVER`	An array of information about the Web server that served the current script
`$_SESSION`	An array of session variables that are available to the current script
`$GLOBALS`	An array of references to all variables that are defined with global scope

Using Autoglobals (continued)

- Use the `global` keyword to declare a global variable within the scope of a function
- Use the `$GLOBALS` autoglobal to refer to the global version of a variable from inside a function
- `$_GET` is the default method for submitting a form
- `$_GET` and `$_POST` allow you to access the values of forms that are submitted to a PHP script

Using Autoglobals (continued)

- $_GET appends form data as one long string to the URL specified by the action attribute
- $_POST sends form data as a transmission separate from the URL specified by the action attribute

Making Decisions

- **Decision making** or **flow control** is the process of determining the order in which statements execute in a program
- The special types of PHP statements used for making decisions are called **decision-making statements** or **decision-making structures**

PHP Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.
  - **if** statement - use this statement to execute some code only if a specified condition is true
  - **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
  - **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
  - **switch** statement - use this statement to select one of many blocks of code to be executed

`if` Statements

- Used to execute specific programming code if the evaluation of a conditional expression returns a value of **true**
- The syntax for a simple `if` statement is:

```
if (conditional expression)
        statement;
```

`if` Statements (continued)

- Contains three parts:
  - the keyword `if`
  - a conditional expression enclosed within parentheses
  - the executable statements
- A **command block** is a group of statements contained within a set of braces
- Each command block must have an opening brace ({) and a closing brace (})

if Statements (continued)

```
$ExampleVar = 5;
if ($ExampleVar == 5) {   // CONDITION EVALUATES TO 'TRUE'
   echo “<p>The condition evaluates to true.</p>”;
   echo '<p>$ExampleVar is equal to ', “$ExampleVar.</p>”;
   echo “<p>Each of these lines will be printed.</p>”;
}
echo “<p>This statement always executes after the if
statement.</p>”;
```

PHP If Statements

- If the condition is evaluated and found true, the statement or the statements immediately following the condition are executed.

If(expression)
Statement

```
<?php
    $minutes = 31;
    if($minutes > 30) {
      echo "Your time is up!<br>";
      echo "Please get out of the pool.";
    }
  ?>
```

```
<?php
$x=1;
if($x==1)
print'$x isequalto1';
?>
```

```
<?php
$x=1;
if ($x == 1)
 {
   print '$x is equal to 1 <br>';
   $x++;
   print 'now $x is equal to 2';
 }
?>
```

```
<?php
$x=1;
if ($x == 1) print '$x is equal to 1 <br>';
if ($x == 1) { print '$x is equal to 1<br>'; }
if ($x == 1) {
print '$x is equal to 1<br>'; }
?>
```

`if...else` Statements

- An `if` statement that includes an `else` clause is called an **`if...else` statement**
- An `else` clause executes when the condition in an `if...else` statement evaluates to **false**
- The syntax for an `if...else` statement is:

```
if (conditional expression)
     statement;
else
     statement;
```

`if...else` Statements (continued)

- An `if` statement can be constructed without the `else` clause
- The `else` clause can only be used with an `if` statement

```
$Today = “Tuesday”;
if ($Today == “Monday”)
     echo “<p>Today is Monday</p>”;
else
     echo “<p>Today is not Monday</p>”;
```

Nested if and if...else Statements

- When one decision-making statement is contained within another decision-making statement, they are referred to as nested **decision-making structures**

```
if ($_GET[“SalesTotal”] > 50)
     if ($_GET[“SalesTotal”] < 100)
          echo “<p>The sales total is between 50 and 100.</p>”;
```

PHP If-else Statements

- If-else is used to make two types of decision, such as true and false, based on a condition.
- When the condition is true, the if statement is executed and when the condition is false the else statement is executed

```
If (condition)
{
      Statement_1
}
Else
{
      Statement_2
 }
```

```
<?php
        $temperature = 66;
        if ($temperature < 32 || $temperature > 100)
{
          echo "Better stay inside today.";
        }
        else {
          echo "Nice day outside.";
        }
       ?>
```

```
<?php
$a=34;
$b=23;
If($a>$b}
{
     Echo “a is greater than b”;
}
Else
      Echo “a is not greater than b”
}
?>
```

PHP If-elseif-else Statements

- If-else is used to make two types of decision, such as true and false, based on a condition.
- When the condition is true, the if statement is executed and when the condition is false the else statement is executed

```
If (condition)
{
            Statement_1
}
Elseif(condition_2)
{
            Statement_2
 }.......
Elseif(condition_n-1)
}
else{ststement_n
}
```

```
<?php
$result = 70;
if ($result >= 75)
{
   echo "Passed -> Grade A <br />";
}
elseif ($result >= 60)
{
   echo "Passed-> Grade B <br />";
}
elseif ($result >= 45)
{
   echo "Passed -> Grade C <br />";
}
else
{
 echo "Failed <br />";
}
?>
```

`switch` Statements

- Controls program flow by executing a specific set of statements depending on the value of an expression
- Compares the value of an expression to a value contained within a special statement called a **case label**
- A **case label** is a specific value that contains one or more statements that execute if the value of the case label matches the value of the switch statement's expression

`switch` Statements (continued)

- Consists of the following components:
  - The `switch` keyword
  - An expression
  - An opening brace
  - A `case` label
  - The executable statements
  - The `break` keyword
  - A default label
  - A closing brace

`switch` Statements (continued)

- The syntax for the `switch` statement is:

```
Switch (expression) {
        case label:
              statement(s);
              break;
        case label:
              statement(s);
              break;
        ...
        default:
              statement(s);
}
```

`switch` Statements (continued)

- A `case` label consists of:
  - The keyword case
  - A literal value or variable name
  - A colon
- A `case` label can be followed by a single statement or multiple statements
- Multiple statements for a `case` label do not need to be enclosed within a command block

`switch` Statements (continued)

- The **`default` label** contains statements that execute when the value returned by the `switch` statement expression does not match a `case` label
- A `default` label consists of the keyword `default` followed by a colon

PHP the Switch Statements

- A switch statement allows a program to evaluate a condition and match the condition's value to the statement s given in case labels show in the following syntax

```
Switch (condition)
{Case label_1;
  Statements_2
  Break;
Case label_2;
  Statements_22
 Break;
….
Default:
 Statements_n
Break;
}
```

```
<?php
$flower = "rose";
switch ($flower)
{  case "rose" :
     echo $flower." costs $2.50";
     break;
  case "daisy" :
     echo $flower." costs $1.25";
     break;
  case "lily" :
     echo $flower." costs $1.50";
     break;
  default :
     echo "There is no such flower in our shop";
     break;
}
?>
```

Repeating Code

- A **loop statement** is a control structure that repeatedly executes a statement or a series of statements while a specific condition is true or until a specific condition becomes true
- There are four types of loop statements:
  - `while` statements
  - `do...while` statements
  - `for` statements
  - `foreach` statements

PHP Looping Statements

- In programming, it is often necessary to repeat the same block of statements a given number of times or until certain conditions are fulfilled.
- This can be accomplished by the looping statements.
- PHP has four types of looping statements
  - The while loop
  - The do-while loop
  - The for loop
  - The foreach loop

PHP the While loop

- While loops executes statements repeatedly, as long as the condition is evaluated and found true.

While (condition)

{

Statement;

}

```
$Count = 10;
while ($Count > 0) {
               echo
"$Count<br />";
               --$Count;
}
echo "<p>We have
liftoff.</p>";
```

```
$Count = 1;
while ($Count <= 5) {
               echo "$Count<br />";
               ++$Count;
}
echo "<p>You have printed 5 numbers.</p>";
```

```
<?php
$i=1;
while($i<=5)
{
echo "The number is ".$i."<br />";
$i++;
}
?>
```

```
<?php
      $variable = 1;

      while ($variable < 10){
         echo "Now \$variable holds: ", $variable, "<br>";
         $variable++;
      }
?>
```

PHP the do-While loop

- Do-while loops executes the block of statements at least once and then checks the condition, and repeats the loop if condition is true.

```
Do
{
Statement;
}
While (condition);
```

```
<?php
$i=1;
do
{
$i++;
echo "The number is".$i."<br />";
}
while ($i<=5);
?>
```

```
<?php
    $variable = 1;

    do {
      echo "Now \$variable holds: ", $variable, "<br>";
      $variable++;
    }
    while ($variable < 10)
  ?>
```

PHP the for loop

The for loop is used when you know how many times you want to execute a statement or a list of statements.

```
For (initialization; condition; increment/decrement)
{
Statement;
}
```

```
<?php
for ($i=1; $i<=5; $i++)
{
echo "The number is".$i."<br />";
}
?>
```

```
<?php
   for ($loop_counter = 0; $loop_counter < 6; $loop_counter++)
   {
     echo "You're going to see this message six times.<br>";
   }
  ?>
```

PHP the foreach loop

The foreach loop is a variation of the for loop and allows you to interate over elements in an array. There are two variations of the foreach loop.

Foreach (array as a value)
{
Statement;
}

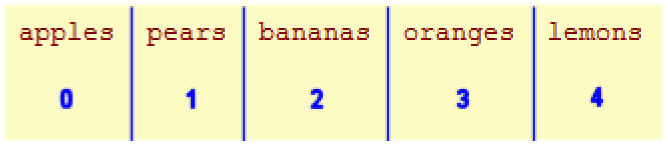
```
<?php
   $arr = array("turkey", "ham", "beef");
   foreach ($arr as $value)
  {
    echo "Current sandwich: $value<br>";
  }
 ?>
```

Foreach (array as key=>value)
{
Statement;
}

```
<?php
$person = array('Name' => 'Ndirangu', 'Age' => 23, 'Address' =>
'Nairobi');
foreach ($person as $key => $value)
{
echo $key." is ".$value."<br />";
}
?>
```

PHP Arrays

- An array is a set of indexed elements where each has its own, unique identification number.
  - An array is a special variable, which can hold more than one value at a time.
- Imagine a list of words separated by commas.
  - It is called a comma-separated list, and it could, for example, look like this: apples, pears, bananas, oranges, lemons
  - Imagine dividing the list at each comma. Next, give each section a unique identification number like this:

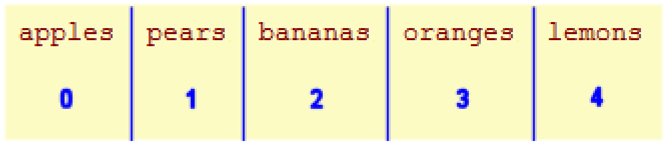


apples	pears	bananas	oranges	lemons
0	1	2	3	4

PHP Arrays

- An array is a set of indexed elements where each has its own, unique identification number.
- Imagine a list of words separated by commas.
  - It is called a comma-separated list, and it could, for example, look like this:

    apples, pears, bananas, oranges, lemons
  - Imagine dividing the list at each comma. Next, give each section a unique identification number like this:



apples	pears	bananas	oranges	lemons
0	1	2	3	4

PHP Arrays

- What you see is an array.
  - Name the array "fruits".
  - The idea is that we can access the array with a number and get a value back, like this:
    - fruits(0) = apples
      fruits(1) = pears
      fruits(2) = bananas
      fruits(3) = oranges
      fruits(4) = lemons

```
<?php
  // create an array with some element
  $cars = array(‘Toyota', ‘Suzuki', ‘Nissan', ‘KIA');
?>
```

```
<?php
        $fruitlist = "apples, pears, bananas, oranges, lemons";
?>
```

PHP Arrays

- To print array elements using print_r() function.
- We have not specified the array keys (indexes) If no key is specified, PHP automatically assigns a numeric key, thus creating a numeric key.

```
<?php
  // create an array with some element
 $cars = array('Toyota', 'Suzuki', 'Nissan', 'KIA');
//print the array contents
 print_r( $cars );
?>
```

PHP Arrays

- Accessing Array elements
  - PHP allows direct access to array values by specifying the array name and the index,

```
<?php
  // create an array with some element
 $cars = array('Toyota', 'Suzuki', 'Nissan', 'KIA');
echo "The Second element of the array is:".$cars[1]
?>
```

PHP Arrays

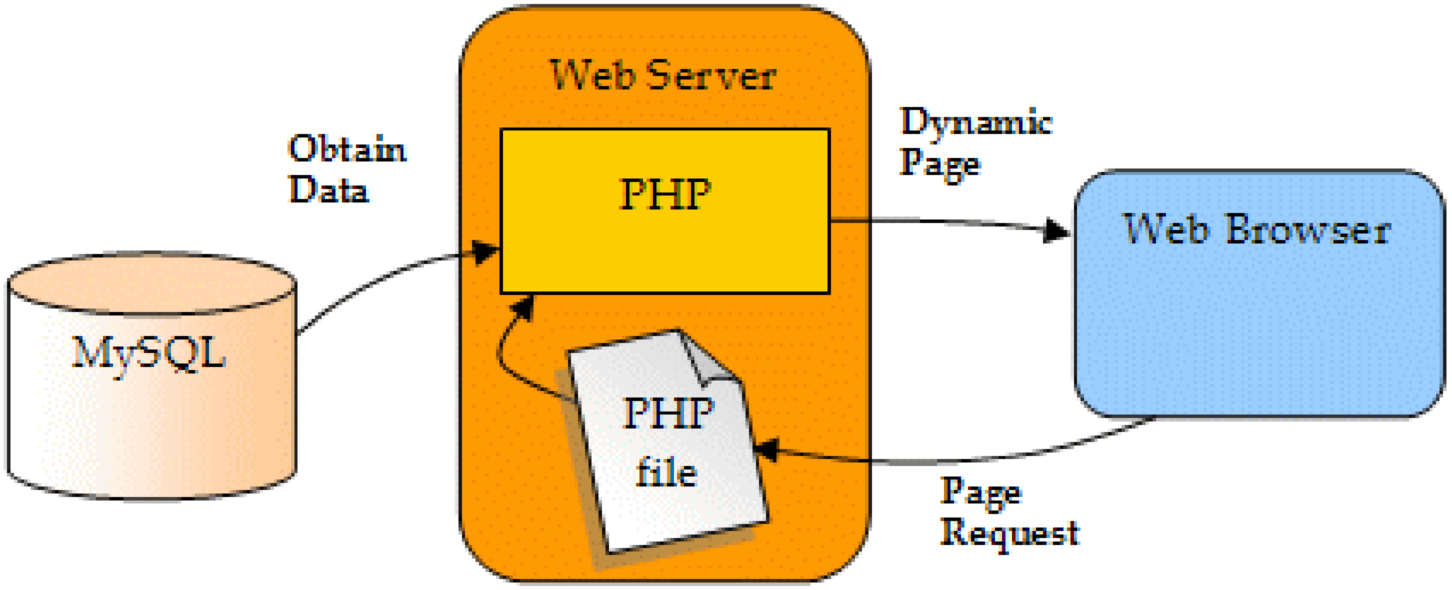
- **Handling arrays with Loops**
  - Arrays are indexed using an array index, and loops use a loop counter
  - By using the loop counter as the array index, you can increment through an entire array I a few lines.

```
<?php
  $actors[0] = "Cary Grant";
  $actors[1] = "Myrna Loy";
  $actors[2] = "Lorne Green";

  for ($loop_index = 0; $loop_index < count($actors); $loop_index++)
  {
    echo "\$actors[$loop_index] = ", $actors[$loop_index], "<br>";
  }
?>
```

```
<?php
$concentrations=array("Forensics","Networks","Applications");
$arrlength=count($concentrations);
for($x=0;$x<$arrlength;$x++)
{
echo $concentrations[$x];
echo "<br>";
}
?>
```

Getting Started with PHP: Working with Databases and MySQL



Introduction to Databases

- A **database** is an ordered collection of information from which a computer program can quickly access information
- Each **row** in a **database table** is called a **record**
- A **record** in a database is a single complete set of related information
- Each **column** in a **database table** is called a field
  - **Fields** are the individual categories of information stored in a record

Create databases and tables

- There are two ways to create databases and tables.
  - Using SQL queries in PHP,
    - The function mysql_query is used to send a query to a MySQL database.
    - The queries are written in the language **S**tructured **Q**uery **L**anguage (SQL). E.g. CREATE DATABASE *database name*
  - The user-friendly tool PhpMyAdmin, which is standard on most web hosts and in XAMPP.

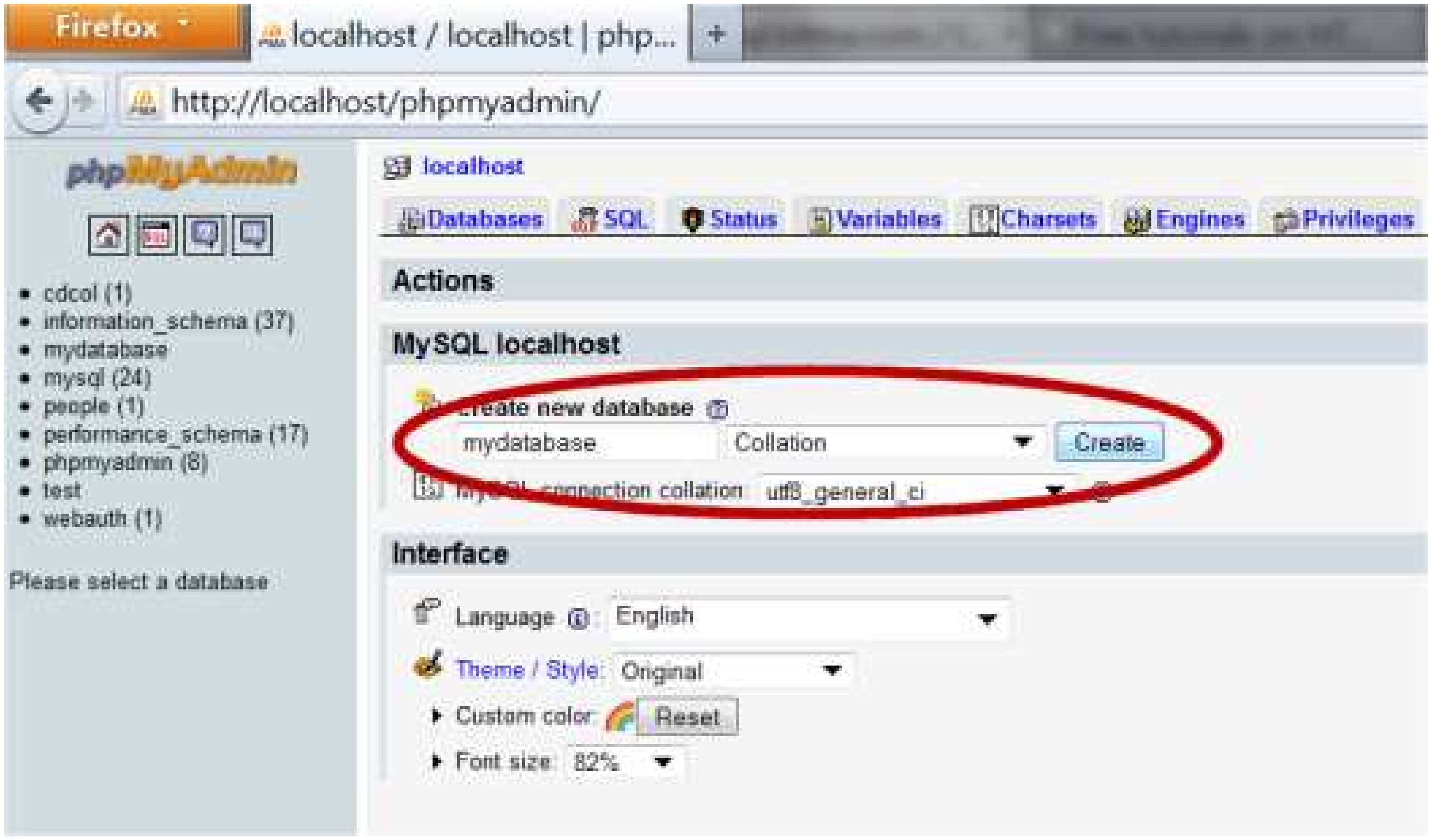
Create database and tables with phpMyAdmin

- It is easier to use phpMyAdmin (or any other MySQL administration tool), which is standard on most web hosts and XAMPP.
  - Start by logging onto phpMyAdmin.
    - Often, the address will be the same as your MySQL server (eg. "http://mysql.myhost.com") and with the same username and password.
  - In XAMPP, the address is http://localhost/phpmyadmin/.
    - When you are logged on, simply type a name for the database and press the button "Create“.

Create database and tables with phpMyAdmin

Std_no	**stud_name**	**major**
451	NGANGA	APT
453	MAGGY	MIS
455	WAKWETU	APT
457	YEBEI	MIS
459	MASSO	APT
461	NJARIA	MIS
463	MASWILI	APT
465	WAIGANJO	MIS

Create database and tables with phpMyAdmin



- cdcol (1)
- information_schema (37)
- mydatabase
- mysql (24)
- people (1)
- performance_schema (17)
- phpmyadmin (8)
- test
- webauth (1)

Please select a database

Actions

MySQL localhost

Create new database

mydatabase | Collation | Create

MySQL connection collation: utf8_general_ci

Interface

Language: English

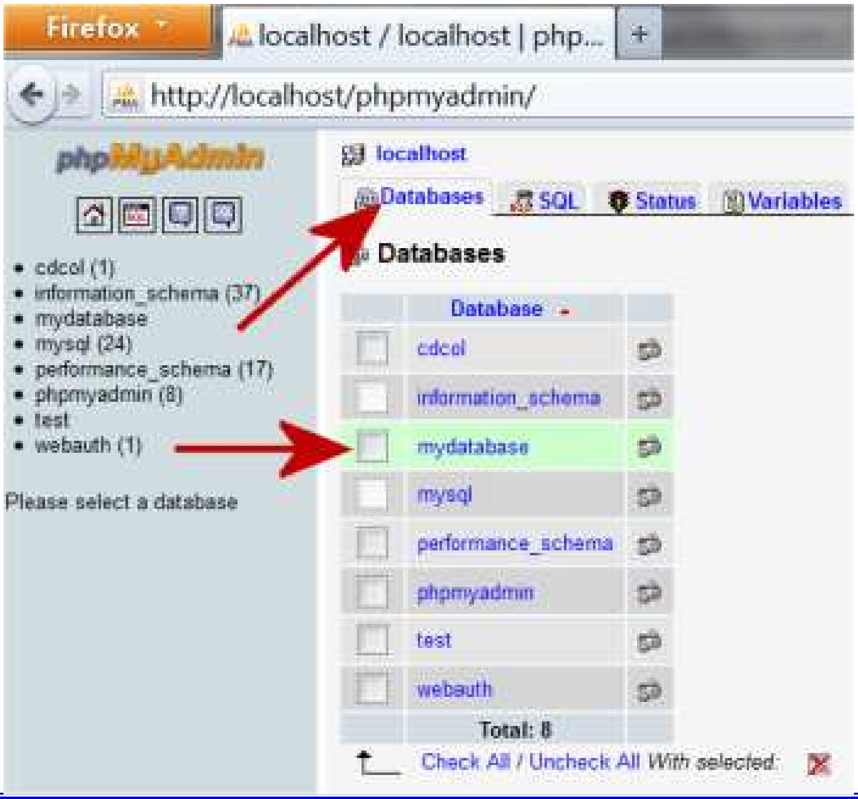
Theme / Style: Original

Custom color: Reset

Font size: 82%

Create database and tables with phpMyAdmin

- At some hosts, it's possible they have already created a database, and you may not have the rights to create more. If that is the case, you obviously just use the assigned database.
- To create a table, click on the tab "Databases" and choose a database by clicking on it:



- cdcol (1)
- information_schema (37)
- mydatabase
- mysql (24)
- performance_schema (17)
- phpmyadmin (8)
- test
- webauth (1)

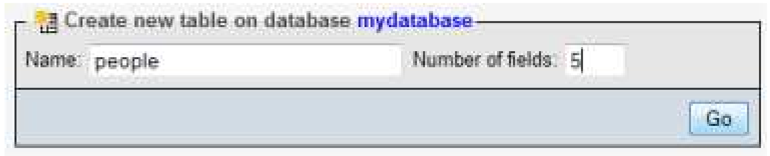
Please select a database

	Database
☐	cdcol
☐	information_schema
☐	mydatabase
☐	mysql
☐	performance_schema
☐	phpmyadmin
☐	test
☐	webauth
	Total: 8

Check All / Uncheck All With selected:

Create database and tables with phpMyAdmin

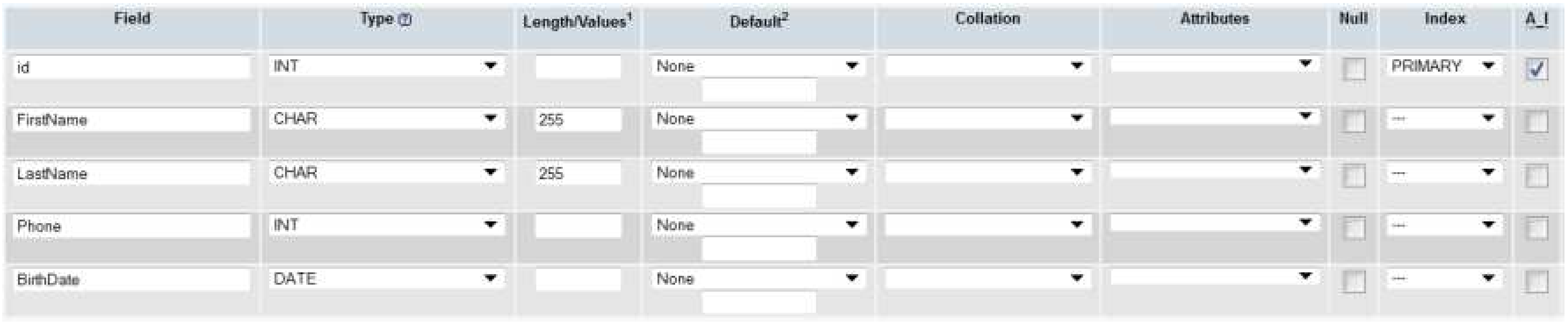
- Then there will be a box titled "Create new table in database", where you type the name of the table and the number of columns and press the button "Go":



Create new table on database mydatabase

Name: people Number of fields: 5

- Then you can name the columns and set the data type.

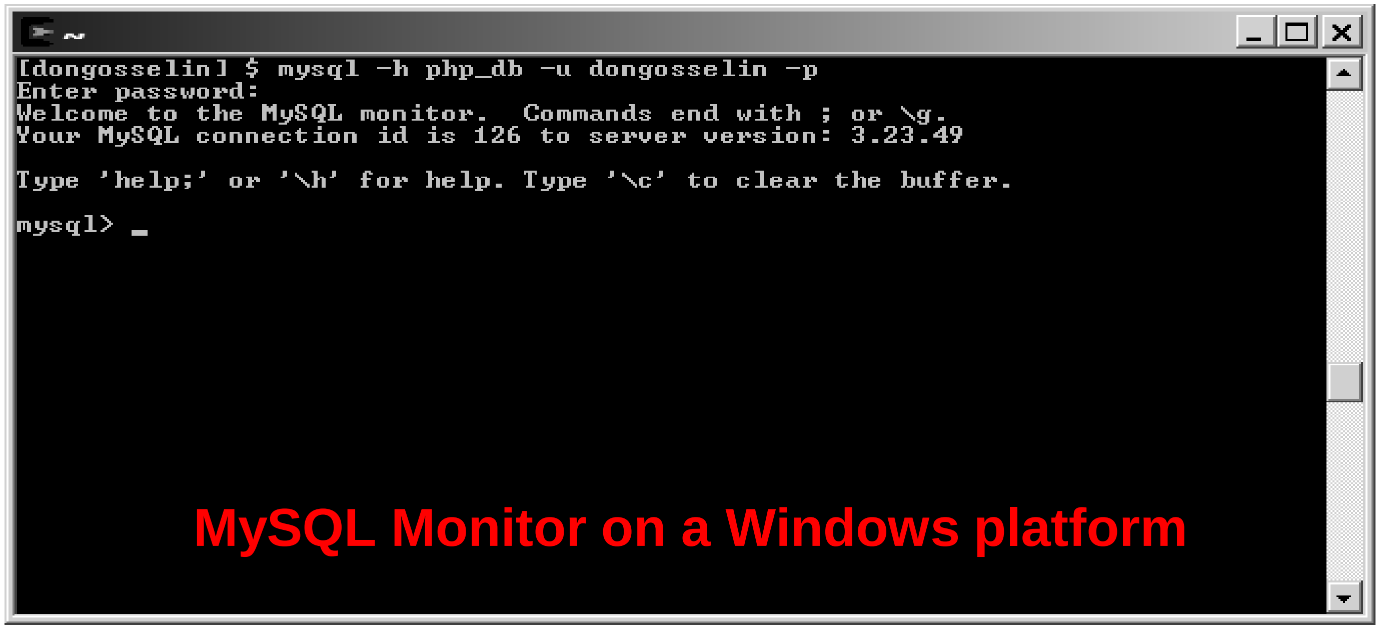


Field	Type	Length/Values¹	Default²	Collation	Attributes	Null	Index	A_I
id	INT		None			☐	PRIMARY	✓
FirstName	CHAR	255	None			☐	---	☐
LastName	CHAR	255	None			☐	---	☐
Phone	INT		None			☐	---	☐
BirthDate	DATE		None			☐	---	☐

- Set PRIMARY KEY and use AUTO_INCREMENT (A_I) if it fits your data.

Getting Started with MySQL

- The **MySQL Monitor** is a command-line program for manipulating MySQL databases
- Connect to the MySQL server using a command-line connect
- Commands are entered at the `mysql->` command prompt in the **console window**



```
[dongosselin] $ mysql -h php_db -u dongosselin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 126 to server version: 3.23.49

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> _
```

MySQL Monitor on a Windows platform

Working with the MySQL Monitor

- At the `mysql>` command prompt terminate the command with a semicolon

```
 mysql> SELECT * FROM students;
```

- Without a semicolon, the MySQL Monitor enters a multiple-line command and changes the prompt to ->

```
 mysql> SELECT * FROM student
    ->
```

- The SQL keywords entered in the MySQL Monitor are not case sensitive

Creating Databases

- Use the CREATE DATABASE statement to create a new database:

```
mysql> CREATE DATABASE vehicle_fleet;[ENTER↵]
```

- To use a new database, select it by executing the USE DATABASE statement

Selecting a Database

- Use the DATABASE( ) function to return the name of the currently active database

```
 mysql> SELECT DATABASE();[ENTER↵]
```

- View the available databases using the SHOW DATABASES statement

```
mysql> SHOW databases;[ENTER↵]
```

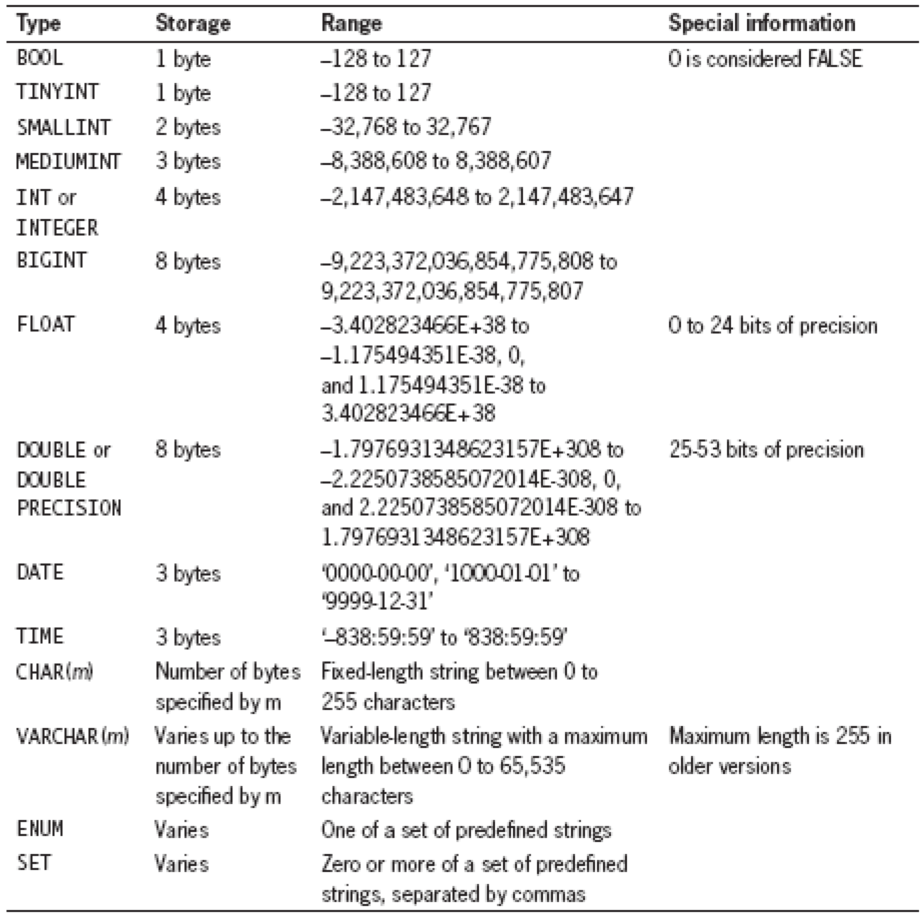
- Use the DROP DATABASE statement to remove all tables and delete a database

```
mysql> DROP DATABASE database;
```

Defining Database Tables

- Data types that are assigned to fields determine how much storage space the computer allocates for the data in the database
- Choose the smallest data type possible for each field

Defining Database Tables



Type	Storage	Range	Special information
BOOL	1 byte	–128 to 127	0 is considered FALSE
TINYINT	1 byte	–128 to 127	
SMALLINT	2 bytes	–32,768 to 32,767	
MEDIUMINT	3 bytes	–8,388,608 to 8,388,607	
INT or INTEGER	4 bytes	–2,147,483,648 to 2,147,483,647	
BIGINT	8 bytes	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	
FLOAT	4 bytes	–3.402823466E+38 to –1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38	0 to 24 bits of precision
DOUBLE or DOUBLE PRECISION	8 bytes	–1.7976931348623157E+308 to –2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308	25-53 bits of precision
DATE	3 bytes	'0000-00-00', '1000-01-01' to '9999-12-31'	
TIME	3 bytes	'–838:59:59' to '838:59:59'	
CHAR(*m*)	Number of bytes specified by m	Fixed-length string between 0 to 255 characters	
VARCHAR(*m*)	Varies up to the number of bytes specified by m	Variable-length string with a maximum length between 0 to 65,535 characters	Maximum length is 255 in older versions
ENUM	Varies	One of a set of predefined strings	
SET	Varies	Zero or more of a set of predefined strings, separated by commas	

Common MySQL Data Types

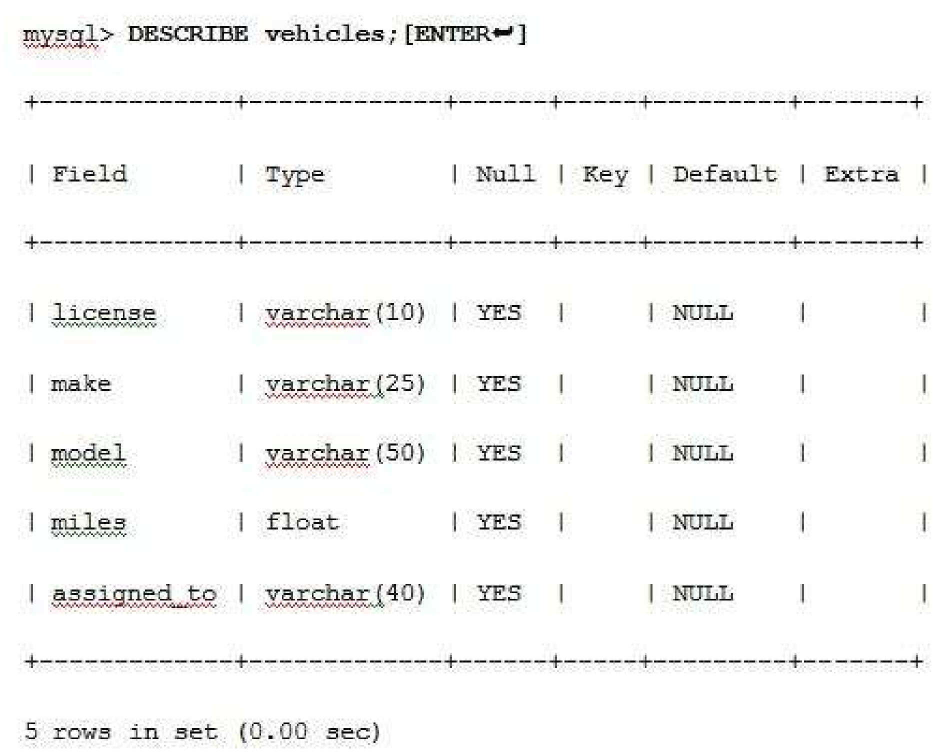
Creating Tables

- Use the CREATE TABLE statement to create a new table and define the column names and data types for each column

```
mysql> CREATE TABLE vehicles
  (license VARCHAR(10), make VARCHAR(25),
   model VARCHAR(50), miles FLOAT,
   assigned_to VARCHAR(40));[ENTER↩]
```

Viewing Table Structure

- Use the DESCRIBE `table_name` statement to view the structure of the table



```
mysql> DESCRIBE vehicles;[ENTER↵]
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| license     | varchar(10) | YES  |     | NULL    |       |
| make        | varchar(25) | YES  |     | NULL    |       |
| model       | varchar(50) | YES  |     | NULL    |       |
| miles       | float       | YES  |     | NULL    |       |
| assigned_to | varchar(40) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Deleting Tables

- Execute the DROP TABLE statement to remove all data and the table definition from a database

```
DROP TABLE table;
```

- In MySQL Monitor, enter the following at the mysql> prompt:

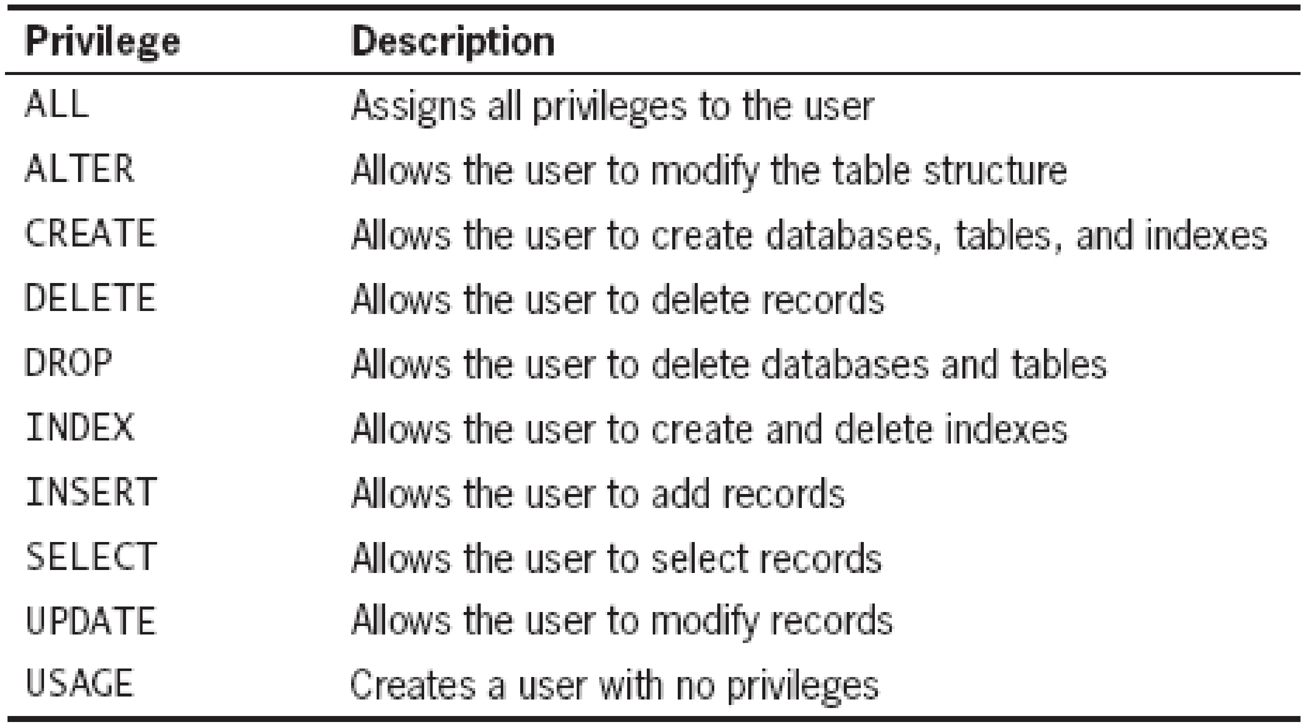
```
mysql> DROP TABLE company_cars;[ENTER↵]
```

- You must be logged in as the `root` user or have DROP privileges to delete a table.

Modifying User Privileges

- `Privileges` are actions and operations a user can perform with a table or a database
- For security purposes, user accounts should only be assigned the minimum necessary privileges to perform given tasks

Modifying User Privileges



Privilege	Description
ALL	Assigns all privileges to the user
ALTER	Allows the user to modify the table structure
CREATE	Allows the user to create databases, tables, and indexes
DELETE	Allows the user to delete records
DROP	Allows the user to delete databases and tables
INDEX	Allows the user to create and delete indexes
INSERT	Allows the user to add records
SELECT	Allows the user to select records
UPDATE	Allows the user to modify records
USAGE	Creates a user with no privileges

Common MySQL Database Privileges

Adding Records

- In MySQL Monitor, enter the following code at the `mysql>` prompt:

```
mysql> INSERT INTO employees (emp_id,
emp_name, title, emp_dept)
  VALUES(666, ‘warui', ‘analyst', 40);
```

Retrieving Records

- Use the SELECT statement to retrieve records from a table:

```
   SELECT criteria FROM table_name;
Select * from employees;
Select * from employees where title='analyst';
```

- Use the asterisk (*) wildcard with the SELECT statement to retrieve all fields from a table
- To return multiple fields, separate field names with a comma
- In MySQL Monitor, enter the following code at the `mysql>` prompt:

```
mysql> SELECT model, mileage FROM
  company_cars;[ENTER↵]
```

Updating Records

- To update records in a table, use the UPDATE statement
- The syntax for the UPDATE statement is:

```
UPDATE table_name
SET column_name=value
WHERE condition;
```

  - The UPDATE keyword specifies the name of the table to update
  - The SET keyword specifies the value to assign to the fields in the records that match the condition in the WHERE keyword
- In MySQL Monitor, enter the following code using the OR keyword at the `mysql>` prompt:

```
mysql> UPDATE company_cars SET mileage=368.2
  WHERE make='Ford’ AND model='Fusion';[ENTER↩]
```

Deleting Records

- Use the DELETE statement to delete records in a table
- The syntax for the DELETE statement is:

```
DELETE FROM table_name
  WHERE condition;
```

- The DELETE statement deletes all records that match the condition
- To delete all the records in a table, leave off the WHERE keyword

Deleting Records

- In MySQL Monitor, enter the following code at the `mysql>` prompt:

```
mysql> DELETE FROM company_cars WHERE
  model_year=2006 AND make='Honda'
  AND model='Accord';[ENTER↩]
```

- To delete all records from a table, omit the WHERE clause

Manipulating MySQL Databases with PHP

Connecting to MySQL with PHP

- PHP has the ability to access and manipulate any database that is ODBC compliant
- PHP includes functionality that allows you to work directly with different types of databases, without going through ODBC
- Accessing Data Using PHP, **three key steps**
  - Connect to the database server
  - Connect to the Database
  - Reading the table

Opening and Closing a MySQL Connection

- Connecting to the database server
  - First, you need to have access to the server where your MySQL database is located. This is done with the function
  - **mysql_connect("localhost", "root", "*****") or die (mysql_error ());**
    - First, you write the location of the database ***(server)***, and then type in the ***username*** and ***password***.
  - **or die(mysql_error())** which, in brief, interrupts the script and writes the error if the connection fails.
  - Keep in mind that it is good practice to close the database connection again when you're finished retrieving or updating data.
    - This is done with the function **mysql_close()**.

Opening and Closing a MySQL Connection

- The syntax for the `mysql_connect()` function is:

```
 <?php
$connection = mysql_connect(“localhost" , "user", "password");
```

- The *host* argument specifies the host name where your MySQL database server is installed
- The *user* and *password* arguments specify a MySQL account name and password

Opening and Closing a MySQL Connection

```
$connection = mysql_connect(“localhost" , "user", "password");
```

- Let’s augment this a little, making it display a message if there is an error, and quit, using the die function

```
$connection = mysql_connect(“localhost" , "user", "password")
or die (“couldn't connect to the server”);
```

Opening and Closing a MySQL Connection

- Connecting to the database
  - The function mysql_select_db() provides the functional interface from which a database can be queried.
  - Use the function mysql_select_db to select the database "APP4050".

mysql_select_db("APP4050", link_identifier) or die(mysql_error(*));

```
<?php
$connection = mysql_connect("localhost" , "user",
"password") or die ("couldn't connect to the server");

$db= mysql_select_db("APP4050", $connection)or die ("could
not Select database");
?>
```

Opening and Closing a MySQL Connection

- Reading the table
  - The function mysql_query() provides the functional from which a database table can be queried.
  - Use the function mysql_query() to select the data from the employees table.

mysql_query(query, link_identifier);

```
<?php
$connection = mysql_connect("localhost" , "user", "password")
or die ("couldn't connect to the server");

$db= mysql_select_db("APP4050", $connection)or die ("could not
Select database");

$query="select * from employees"; // query for getting records

$result=mysql_query($query)
or die(Query Failed:".mysql_error());
?>
```

Opening and Closing a MySQL Connection

- Displaying the table data
  - The function mysql_fetch_array() provides the functional that returns an array corresponding to the current record and can loop over the records using loops
  - Use the function mysql_query() to select the data from the employees table.

mysql_fetch_array(result, result_type);

Opening and Closing a MySQL Connection

```
<?php
$connection = mysql_connect(“localhost" , "user", "password") or
die (“couldn't connect to the server”);

$db= mysql_select_db(“APP4050”, $connection)or die (“could not
Select database”);

$query=“select * from employees”; // query for getting records

$result=mysql_query($query)
or die(Query Failed:”.mysql_error());
Echo “<table border=‘1’”>;
Echo “<tr>”;
echo<“<th>emp_id</th><th>emp_name</th><th>title</th><th>emp_dept</th>”;
Echo “</tr>”;
?>
```

Escape sequences

- Escape sequences, the combination of the escape character \ and a letter, are used to signify that the character after the escape character should be treated specially.
- For example, if you wanted to have the string "And then he said, "That is amazing!", which was true", you would need escape characters because you have double quotes inside double quotes.

\"	Print the next character as a double quote, not a string closer
\'	Print the next character as a single quote, not a string closer
\n	Print a new line character (remember our print statements?)
\t	Print a tab character
\r	Print a carriage return (not used very often)
\$	Print the next character as a dollar, not as part of a variable
\\	Print the next character as a backslash, not an escape character