INFO1-CE9224: Introduction to PHP Programming

Session 2
June 13, 2012

Resources

http://davehauenstein.com/nyu/INFO1-CE9224-2012-Summer

Username: nyuscps
Password: $nyuscps$

Class 2 Agenda

- Review: PHP Language Basics Part 1
- PHP Language Basics Part 2
  - Operators and Expressions
  - Constants
  - Built-in String Functions
- Lab Assignment

Review: PHP Language Basics Part 1

Variables

A variable is a container that holds a certain value.

Naming Variables

- Variables always begin with a dollar sign ($)
- The first character after the $ must be a letter or an underscore (_)
- The remaining characters may be letters, numbers, or underscores
- Variables are case-sensitive

Valid vs Invalid

- $some_variable
- $_someVariable
- $_someVariable2
- $_123

- $1badvariable
- $another-bad-var

PHP Data Types

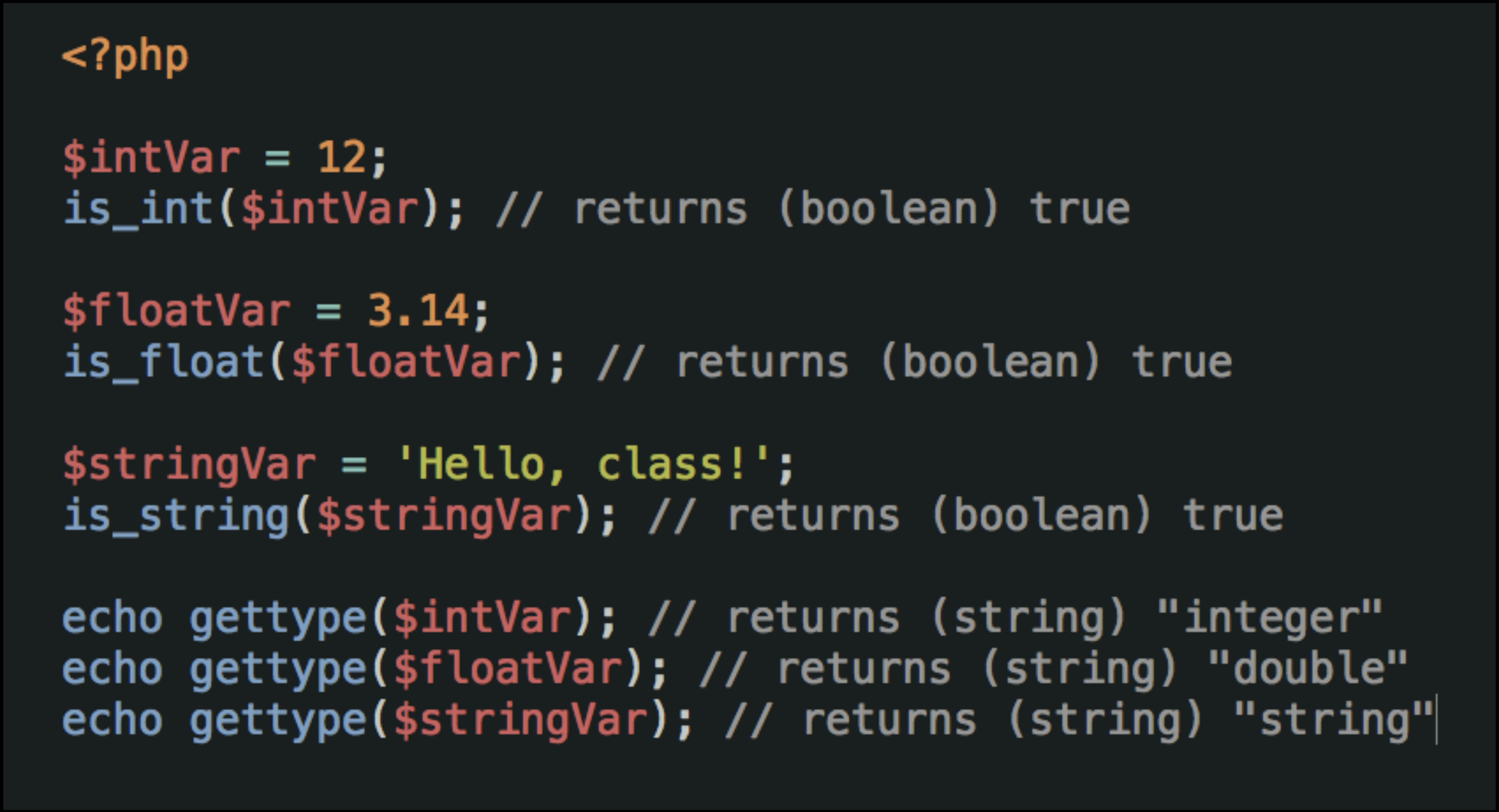
Scalar Data Type	Description	Example
Integer	A whole number	15
Float	A floating-point number	8.23
String	A series of characters	"Hello, World!"
Boolean	Represents either true or false	TRUE

Compound Data Type	Description
Array	An ordered map (contains names or numbers mapped to values
Object	A type that may contain properties and methods

PHP Data Types cont...

Special Data Type	Description
Resource	Contains a reference to an external resource, such as a file or database
null	May only contain null as a value, meaning the variable explicitly does not contain any value

Type Checking

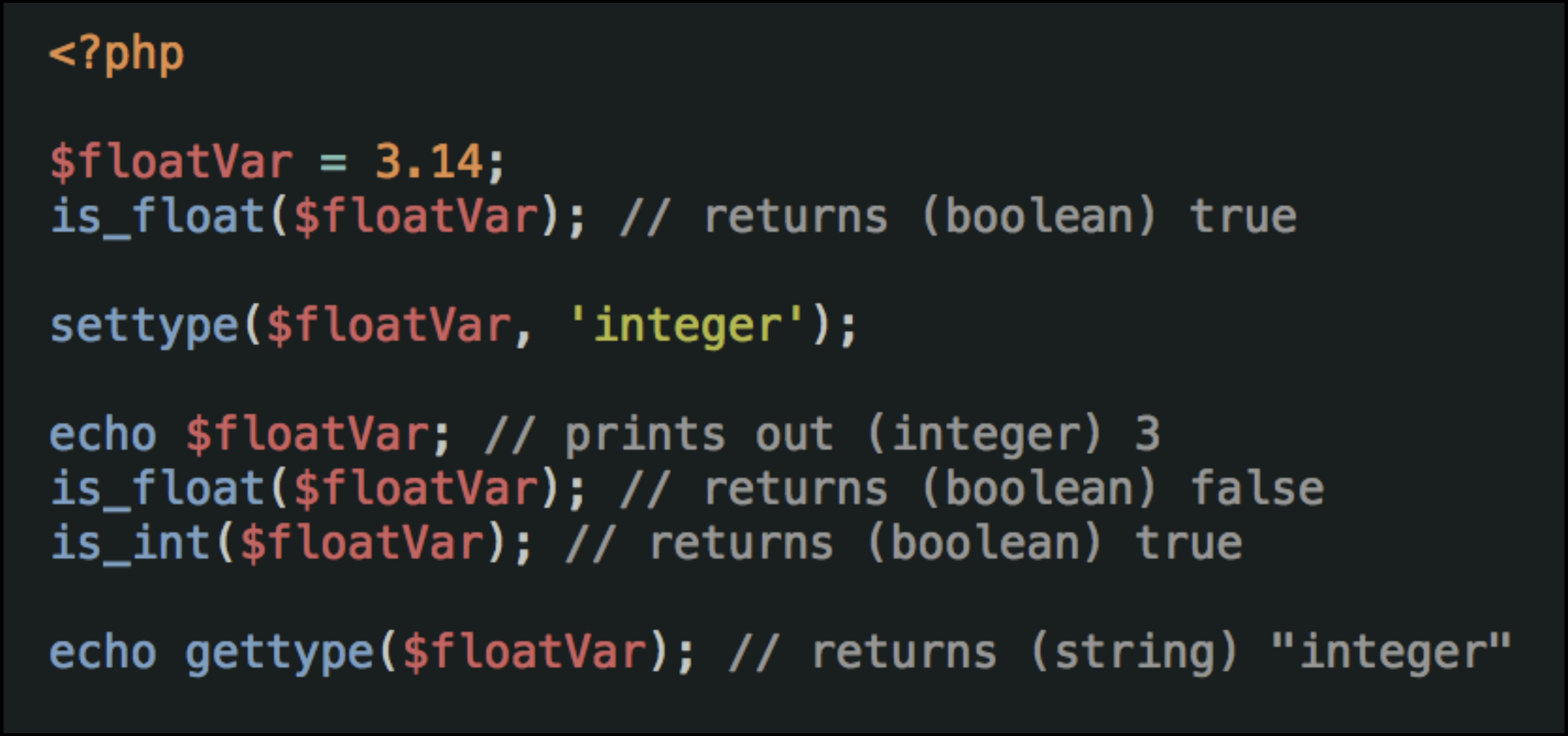


```php
<?php

$intVar = 12;
is_int($intVar); // returns (boolean) true

$floatVar = 3.14;
is_float($floatVar); // returns (boolean) true

$stringVar = 'Hello, class!';
is_string($stringVar); // returns (boolean) true

echo gettype($intVar); // returns (string) "integer"
echo gettype($floatVar); // returns (string) "double"
echo gettype($stringVar); // returns (string) "string"
```

PHP Type Checking

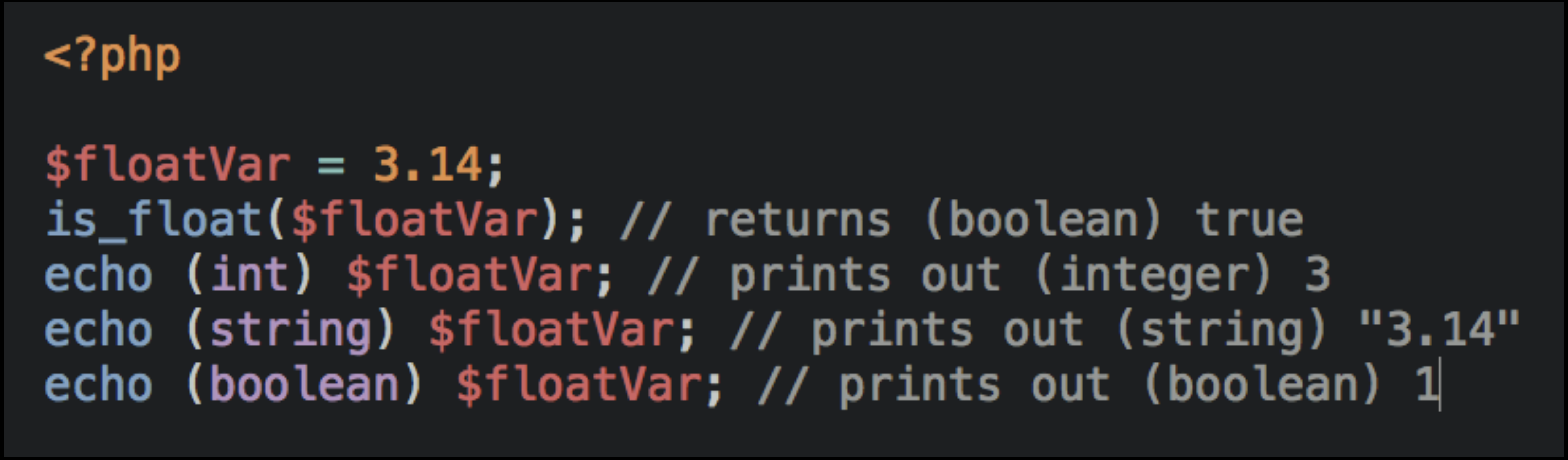
- is_int (*value*)
- is_float (*value*)
- is_string (*value*)
- is_bool (*value*)
- is_array (*value*)
- is_object (*value*)
- is_resource (*value*)
- is_null (*value*)
- gettype (*value*)

Changing a Variable’s Data Type



```
<?php

$floatVar = 3.14;
is_float($floatVar); // returns (boolean) true

settype($floatVar, 'integer');

echo $floatVar; // prints out (integer) 3
is_float($floatVar); // returns (boolean) false
is_int($floatVar); // returns (boolean) true

echo gettype($floatVar); // returns (string) "integer"
```

Type Casting



```php
<?php

$floatVar = 3.14;
is_float($floatVar); // returns (boolean) true
echo (int) $floatVar; // prints out (integer) 3
echo (string) $floatVar; // prints out (string) "3.14"
echo (boolean) $floatVar; // prints out (boolean) 1
```

PHP Language Basics
Part 2

Operators

Manipulate the contents of one or more variables to produce a new value.

```
$x = 10;
$y = 23;
echo $x + $y;   // prints out 33
```

Operator

Operand

Expressions

Anything that evaluates to a value.

```
$x + $y
$x - $y
$a + $b + $c
92
false
gettype( $x )
```

Operator Types

Type	Description
Arithmetic	Perform common arithmetical operations such as addition and subtraction
Assignment	Assign values to variables
Comparison	Compare values in a boolean fashion (true or false is returned)
Error Control	Affect Error Handling
Execution	Cause execution of commands as though they were shell commands
Incrementing / Decrementing	Increment or decrement a variable’s value
Logical	Boolean operators such as *and*, *or*, and *not* that can be used to include or exclude
String	Concatenates strings (only 1 string operator)
Array	Perform operations on arrays

Arithmetic Operators

Operator	Example
+ (addition)	5 + 4 = 9
- (subtraction)	9 - 5 = 4
* (multiplication)	5 * 4 = 20
/ (division)	20 / 4 = 5
% (modulus)	20 % 4 = 0 20 % 3 = 2

Assignment Operators

```
$firstName = “Jessica”;
```

Assignment Operators

```
$firstName = “Jessica”;
```

“Jessica”

Assignment Operators

$firstName = “Jessica”;

“Jessica”

$name = $firstName = “Jessica”;

Assignment Operators

The assignment operator not only performs the assignment, but produces a value as well.

Combined Assignment

```
$someNum = 3;
$otherNum = 7;
$someNum = $someNum + $otherNum;
```

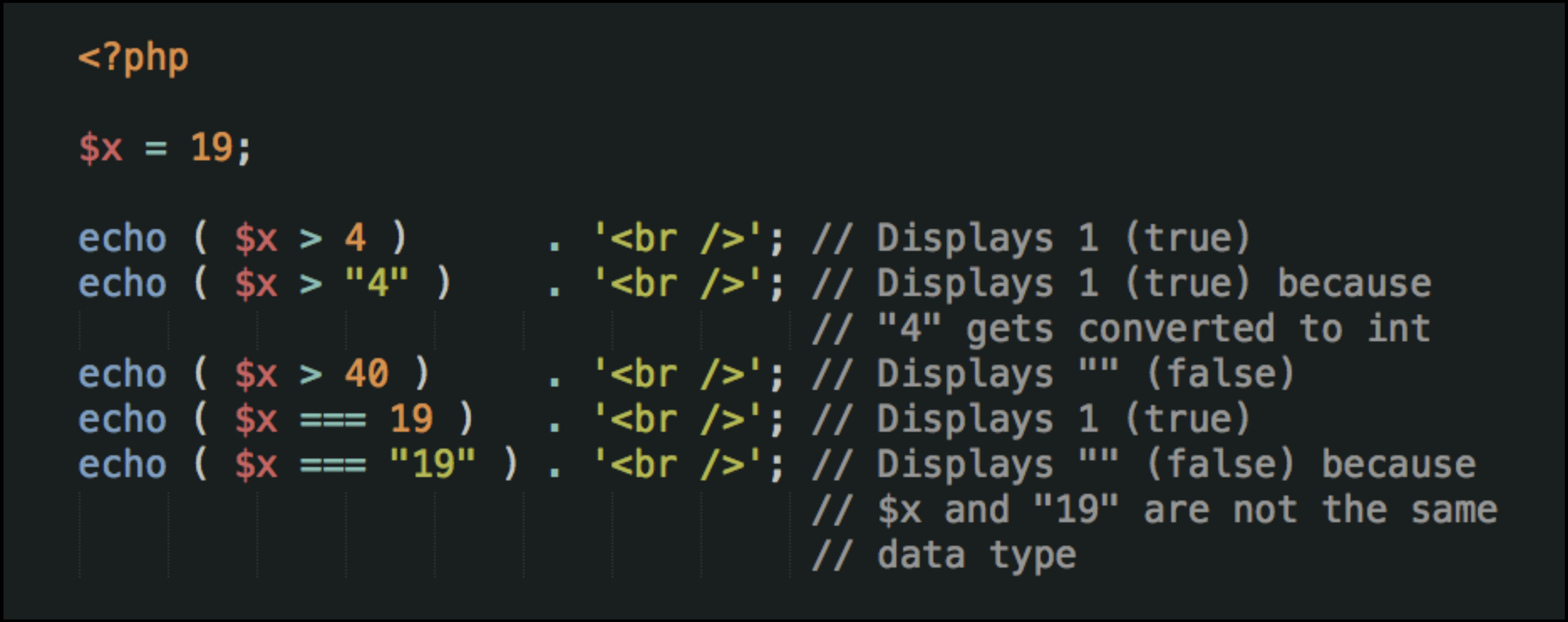
Combined Assignment

```
$someNum = 3;
$otherNum = 7;
$someNum = $someNum + $otherNum;

$someNum += $otherNum;
```

Comparison Operators

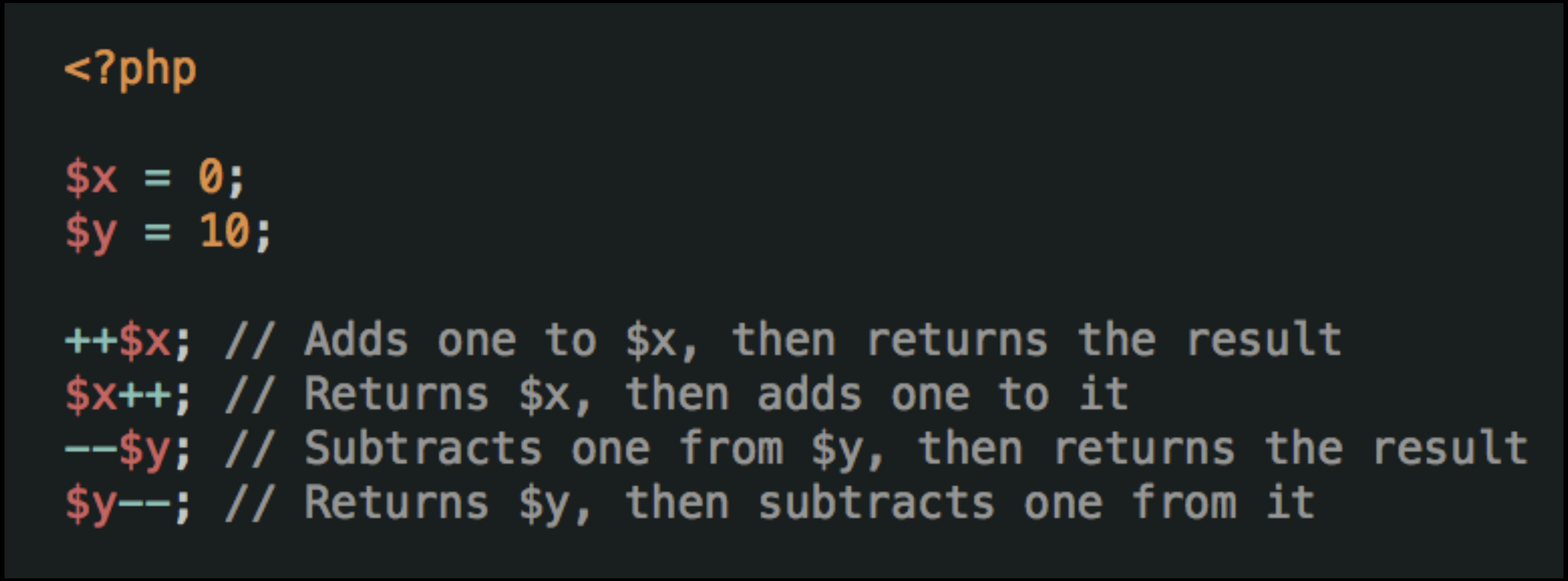
Operator	Example	Result
== (equal)	$x == $y	true if $x equals $y; false otherwise
!= or <> (not equal)	$x != $y	true if $x does not equal $y; false otherwise
=== (identical)	$x === $y	true if $x equals $y and they are of the same type; false otherwise
!== (not identical)	$x !== $y	true if $x does not equal $y or they are of the same type; false otherwise
< (less than)	$x < $y	true if $x is less than $y; false otherwise
> (greater than)	$x > $y	true if $x is greater than $y; false otherwise
<= (less than or equal to)	$x <= $y	true if $x is less than or equal to $y; false otherwise
>= (greater than or equal to)	$x >= $y	true if $x is greater than or equal to $y; false otherwise

Comparison Operators



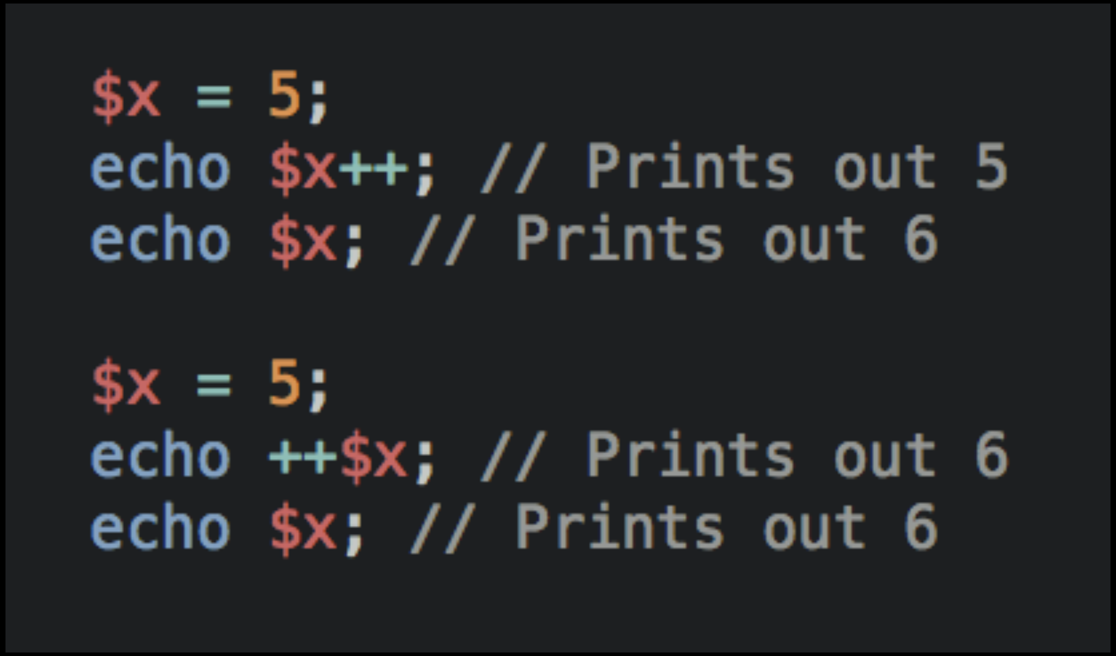
```php
<?php

$x = 19;

echo ( $x > 4 )       . '<br />'; // Displays 1 (true)
echo ( $x > "4" )     . '<br />'; // Displays 1 (true) because
                                  // "4" gets converted to int
echo ( $x > 40 )      . '<br />'; // Displays "" (false)
echo ( $x === 19 )    . '<br />'; // Displays 1 (true)
echo ( $x === "19" )  . '<br />'; // Displays "" (false) because
                                  // $x and "19" are not the same
                                  // data type
```

Incrementing/ Decrementing



```php
<?php

$x = 0;
$y = 10;

++$x; // Adds one to $x, then returns the result
$x++; // Returns $x, then adds one to it
--$y; // Subtracts one from $y, then returns the result
$y--; // Returns $y, then subtracts one from it
```

Incrementing/ Decrementing



```
$x = 5;
echo $x++; // Prints out 5
echo $x; // Prints out 6

$x = 5;
echo ++$x; // Prints out 6
echo $x; // Prints out 6
```

Logical Operators

<u>True</u>

- 1
- 1 == 1
- 5 > 2
- “hello” != “goodbye”

<u>False</u>

- 5 < 2
- gettype(3) == “array”
- “hello” == “goodbye”
- 23 === “23”

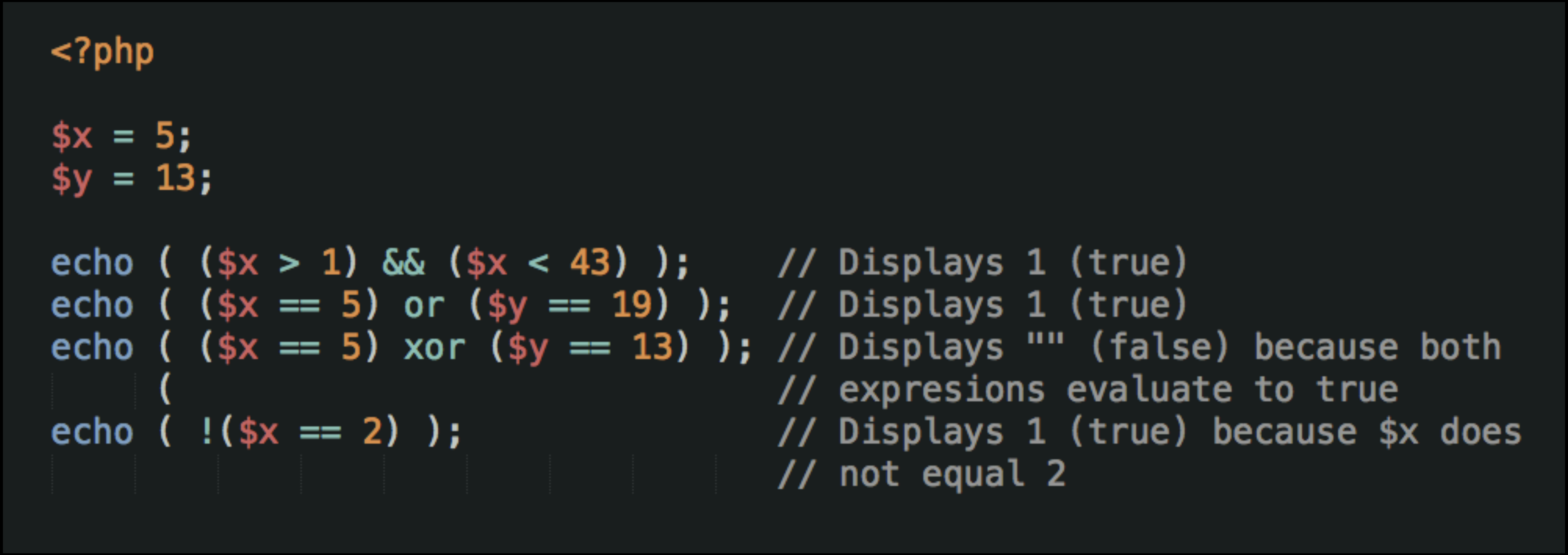
Additional False Values

- the literal value *false*
- The integer zero (0)
- The float zero (0.0)
- An empty string (“ “)
- The string zero (“0”)
- An array with zero elements
- The *null* type
- A SimpleXML object created from an empty XML tag

Logical Operators

Operator	Example	Result
&& (and)	$x && $y	true if both $x and $y evaluate to true; false otherwise
and	$x and $y	true if both $x and $y evaluate to true; false otherwise
|| (or)	$x || $y	true if either $x or $y evaluates to true; false otherwise
or	$x or $y	true if either $x or $y evaluates to true; false otherwise
xor	$x xor $y	true if $x or $y (but not both) evaluates to true; false otherwise
! (not)	!$x	true if $x is false; false if $x is true

Logical Operators



```php
<?php

$x = 5;
$y = 13;

echo ( ($x > 1) && ($x < 43) );    // Displays 1 (true)
echo ( ($x == 5) or ($y == 19) );  // Displays 1 (true)
echo ( ($x == 5) xor ($y == 13) ); // Displays "" (false) because both
     (                             // expresions evaluate to true
echo ( !($x == 2) );               // Displays 1 (true) because $x does
                                   // not equal 2
```

Logical Operators

Short Circuiting

```
$a = ( false && foo() );

$b = ( true || foo() );
```

String Operators

Concatenation operator...

. (dot)

String Operators

Concatenation operator...

. (dot)

```
echo “The weather “ .“is quite chilly”;
```

```
echo “The weather “ .“is quite chilly ” .“today”;
```

String Operators

```
$tempF = 451;
echo "Books catch fire at " . ( (5/9) * ($tempF-32) ) ." degrees C.";
```

Operator Precedence

4 + 5 * 8

Is the answer 72?
OR
Is the answer 44?

Operator Precedence

4 + 5 * 8

Is the answer 72?
OR
Is the answer 44?

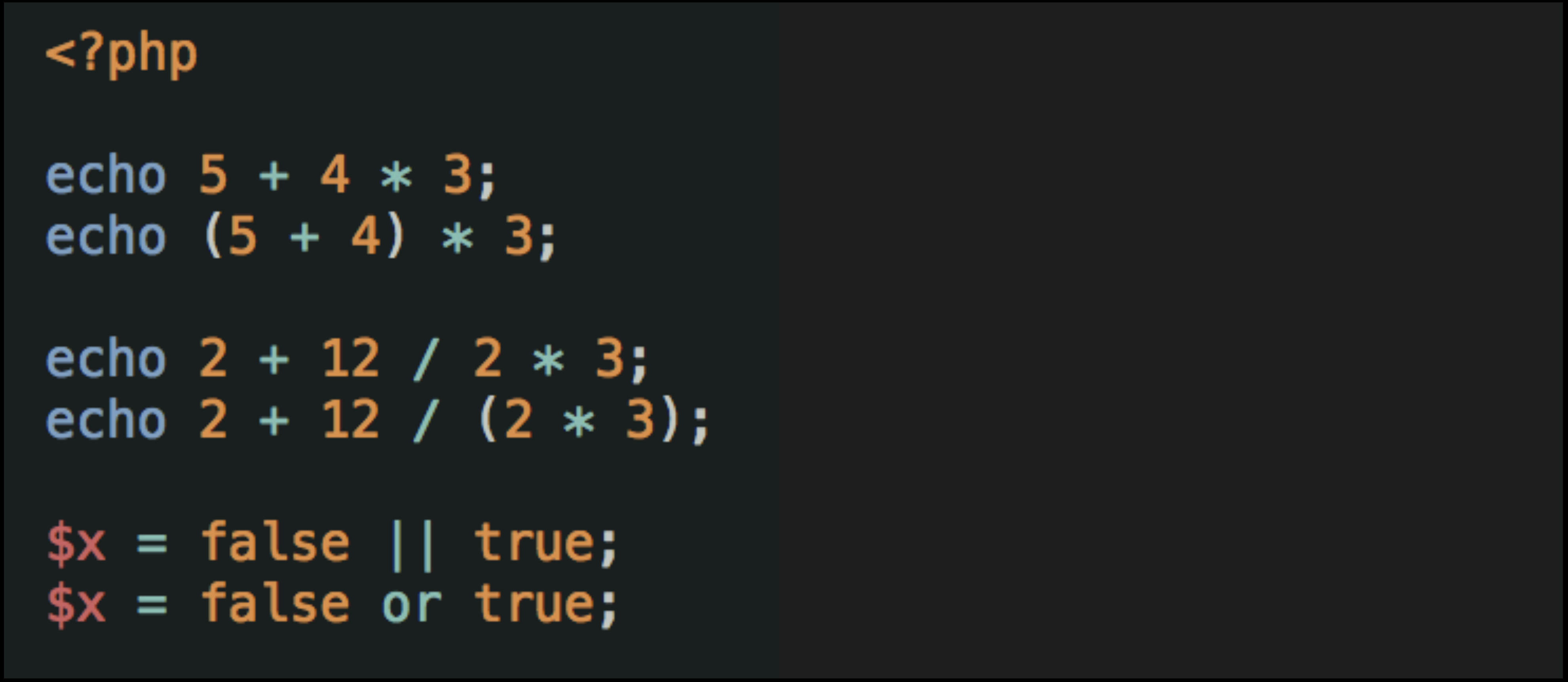
The answer is 44!

PHP Operator Precedence

Precedence of PHP Operators (Higest First)
++ -- (increment / decrement)
(int) (float) (string) (array) (object) (bool) ***(casting)***
! (not)
* / % (arithmetic)
+ - . (arithmetic)
< <= > >= <> (comparison)
== !== === != (comparison)
&& (and)
|| (or)
= += -= *= /= .= %= (assignment)
and
xor
or

Operator Precedence

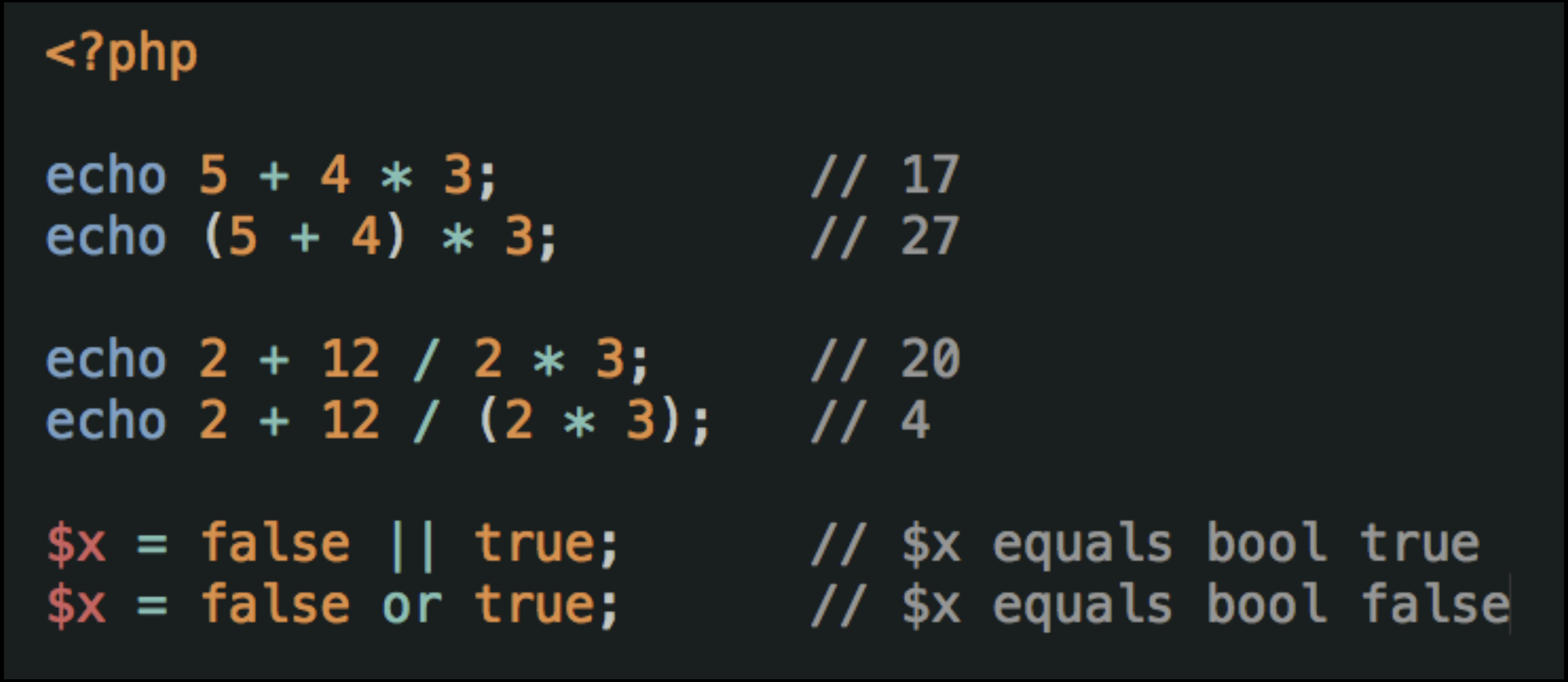
Examples



```php
<?php

echo 5 + 4 * 3;
echo (5 + 4) * 3;

echo 2 + 12 / 2 * 3;
echo 2 + 12 / (2 * 3);

$x = false || true;
$x = false or true;
```

Operator Precedence

Examples

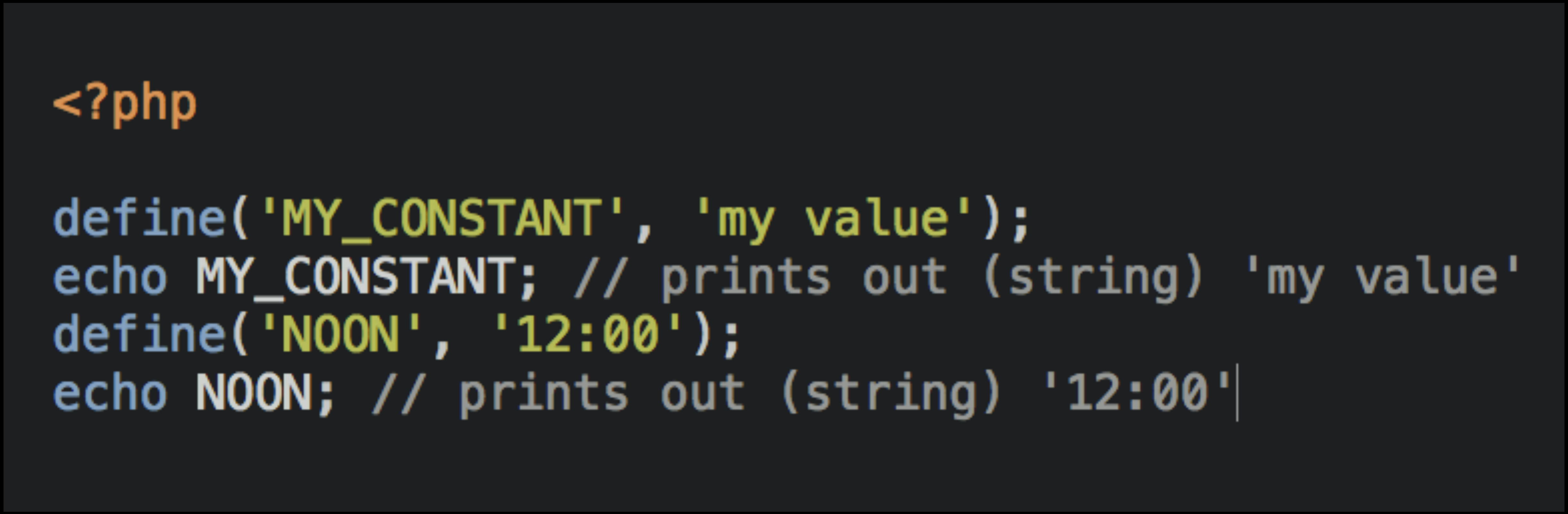


```php
<?php

echo 5 + 4 * 3;           // 17
echo (5 + 4) * 3;         // 27

echo 2 + 12 / 2 * 3;      // 20
echo 2 + 12 / (2 * 3);    // 4

$x = false || true;       // $x equals bool true
$x = false or true;       // $x equals bool false
```

Constants

- Value containers that can never be changed.
- Can only be defined once in a PHP program.
- Names don’t start with $ (dollar sign)
- Standard practice to use ALLCAPS when defining them.
- Can only contain scalar values.
- Can be used anywhere regardless of scope.
- Case sensitive.
- Useful when you want to make sure a value doesn’t ever change.

Constants



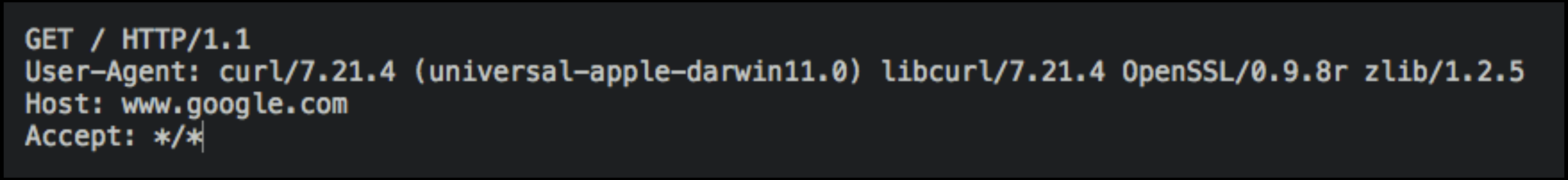
```php
<?php

define('MY_CONSTANT', 'my value');
echo MY_CONSTANT; // prints out (string) 'my value'
define('NOON', '12:00');
echo NOON; // prints out (string) '12:00'
```

Strings

Strings are a sequence of characters:

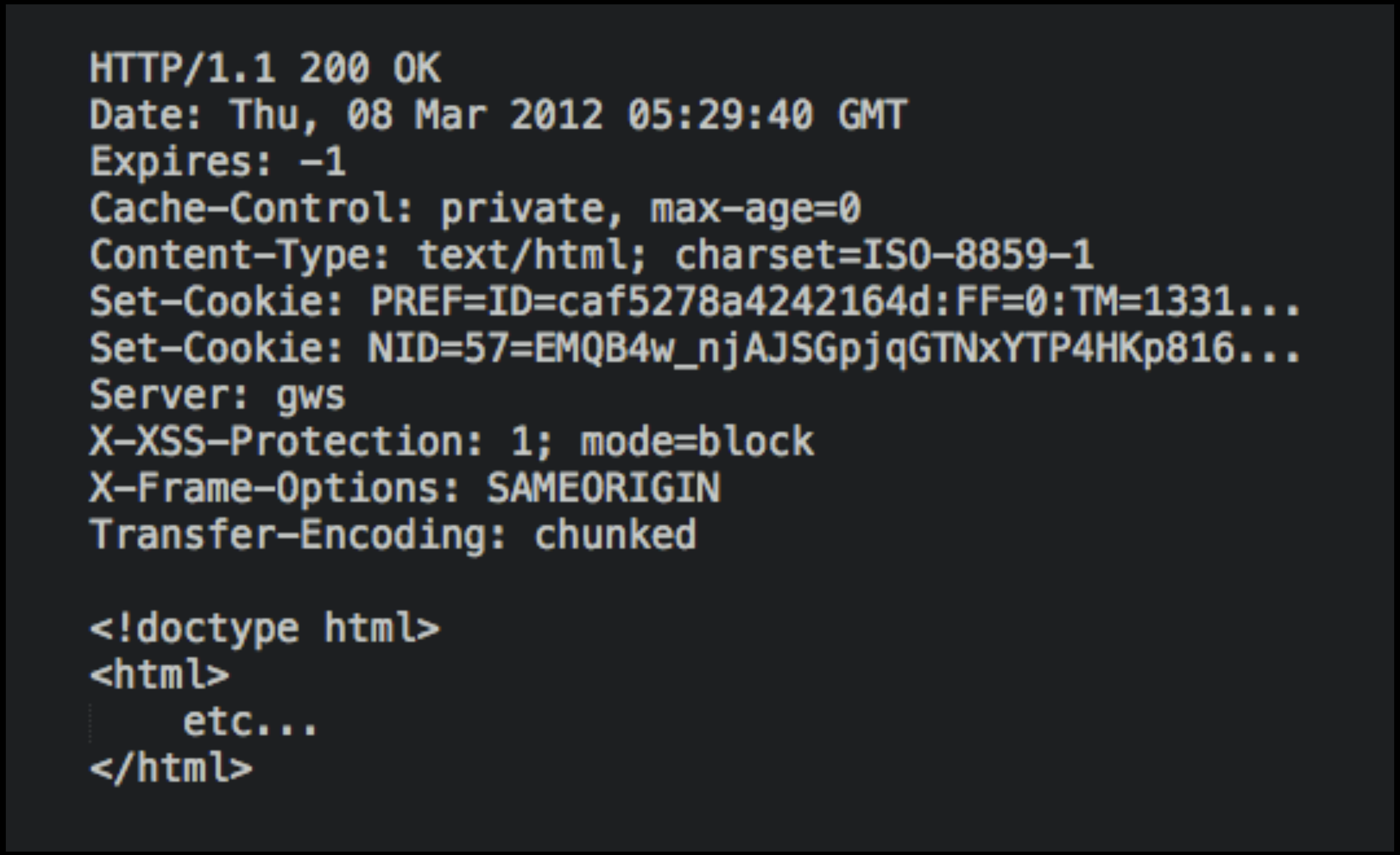
- “abc”
- “Hello, world!”
- “^$()@)(^%)))(_”
- “123456”

HTTP Request



```
GET / HTTP/1.1
User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r zlib/1.2.5
Host: www.google.com
Accept: */*
```

HTTP Response



```
HTTP/1.1 200 OK
Date: Thu, 08 Mar 2012 05:29:40 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: PREF=ID=caf5278a4242164d:FF=0:TM=1331...
Set-Cookie: NID=57=EMQB4w_njAJSGpjqGTNxYTP4HKp816...
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked

<!doctype html>
<html>
    etc...
</html>
```

Creating and Accessing

Simply assign the literal string value to a variable:

- $someString = “Hello, world!”;   // using double quotes
- $someString = ‘Hello again, world!’;   // using single quotes

Creating and Accessing

Simply assign the literal string value to a variable:

- $someString = “Hello, world!”;   // using double quotes
- $someString = ‘Hello again, world!’;   // using single quotes

What’s the difference?

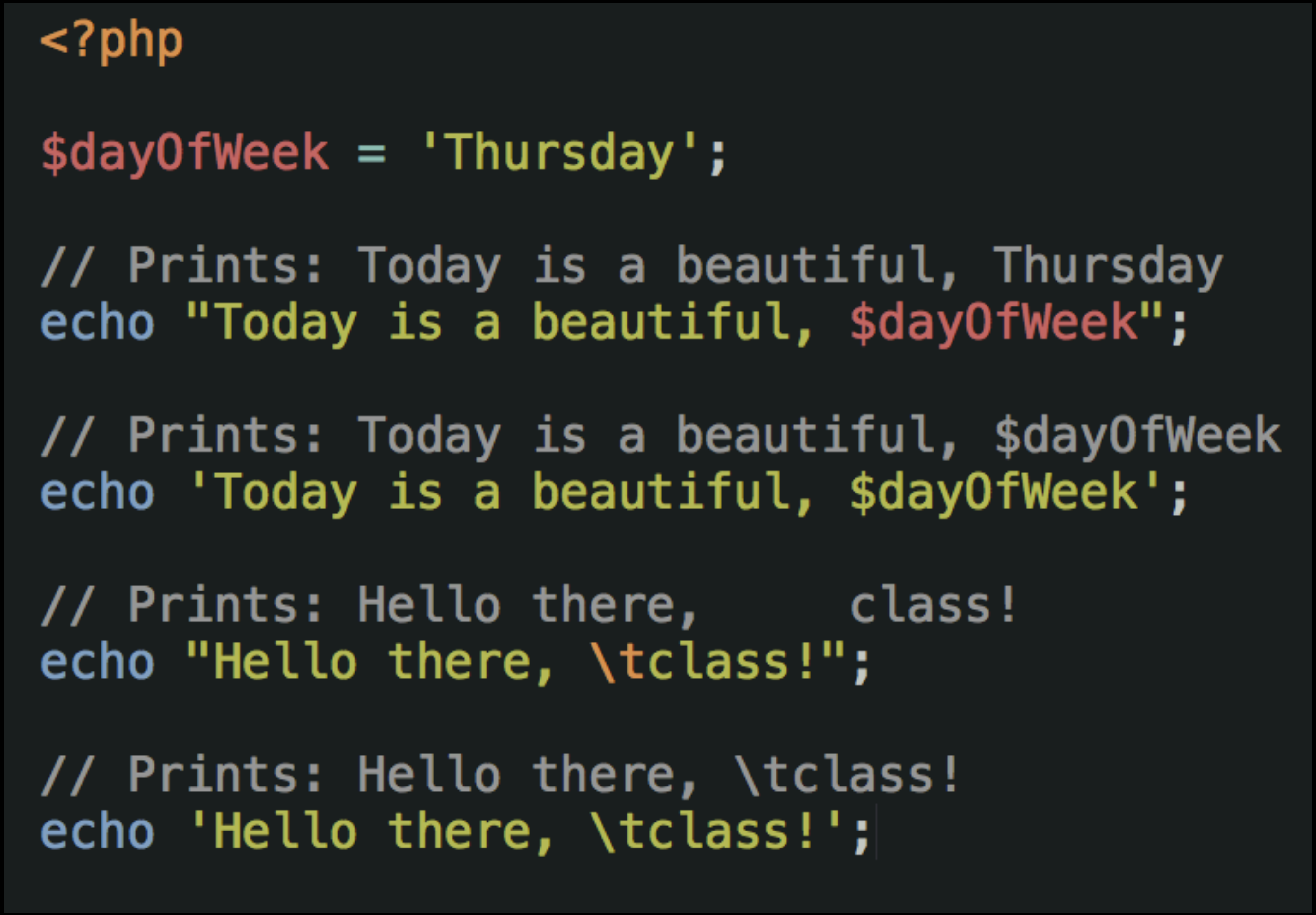
Double quotes...

- Variables enclosed in double quotes are replaced with the value stored in the variable.
- Special characters (escape sequences) can be included by escaping them.

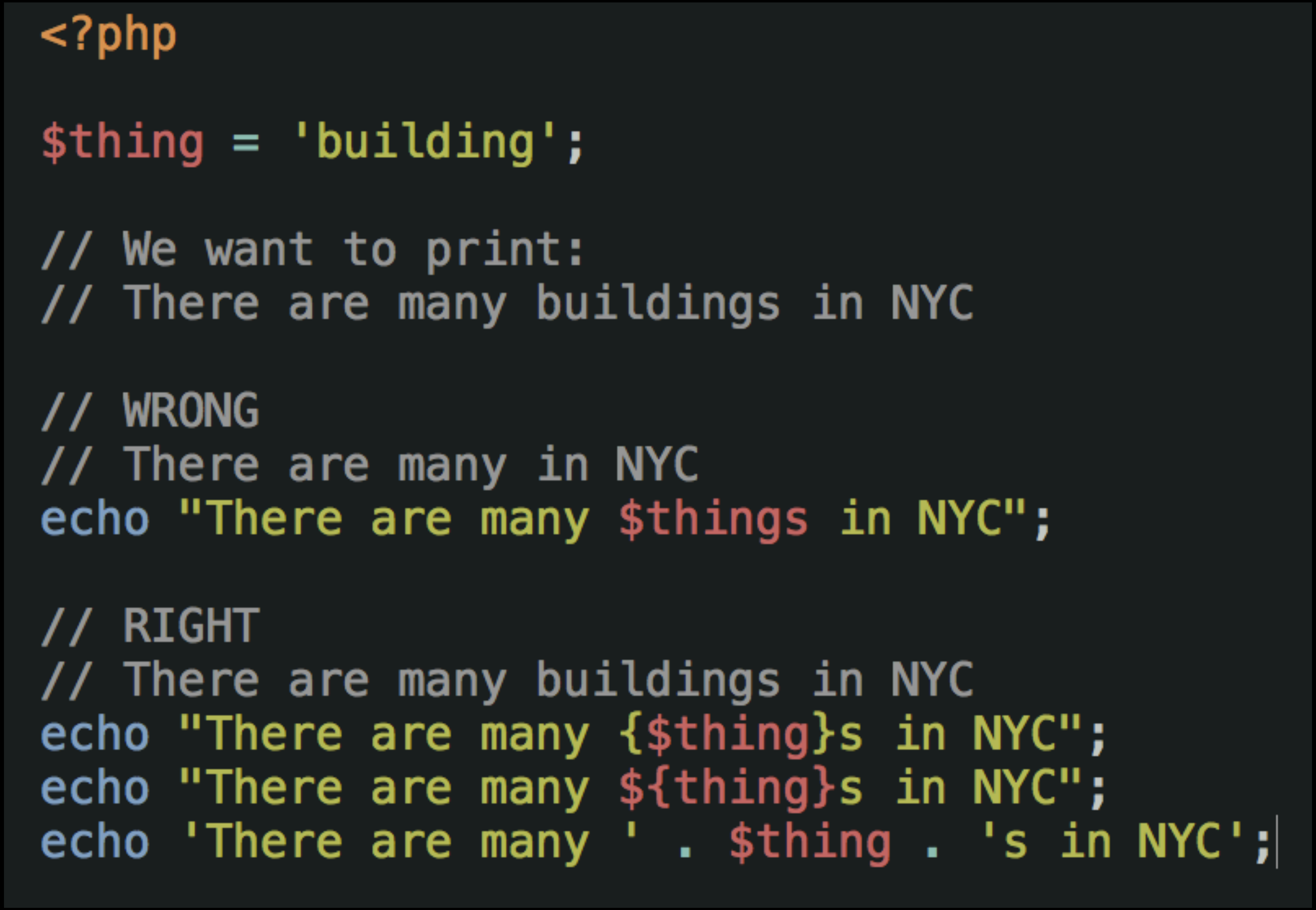
Single quotes...

- PHP treats strings w/in single quotes as literally as they are typed.
- Variables are not replaced with the value stored inside the variable and escape sequences aren't parsed.

Single vs. Double Quotes



```php
<?php

$dayOfWeek = 'Thursday';

// Prints: Today is a beautiful, Thursday
echo "Today is a beautiful, $dayOfWeek";

// Prints: Today is a beautiful, $dayOfWeek
echo 'Today is a beautiful, $dayOfWeek';

// Prints: Hello there,     class!
echo "Hello there, \tclass!";

// Prints: Hello there, \tclass!
echo 'Hello there, \tclass!';
```



```php
<?php

$thing = 'building';

// We want to print:
// There are many buildings in NYC

// WRONG
// There are many in NYC
echo "There are many $things in NYC";

// RIGHT
// There are many buildings in NYC
echo "There are many {$thing}s in NYC";
echo "There are many ${thing}s in NYC";
echo 'There are many ' . $thing . 's in NYC';
```

String Handling and Manipulation functions

Why?

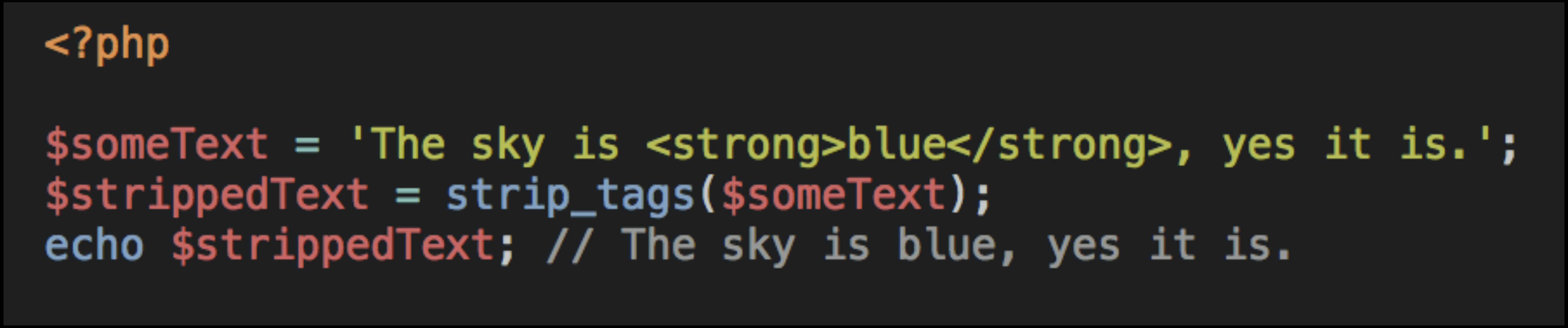
- Format user input for storage or display
- Format numbers or currency
- Search for and extract specific content within a set of data
- Scrape website content
- Parse XML
- Validate data
- Data evaluation, is the content relevant?
- So much more!

Functions

A function is a portion of code within a larger program that performs a specific task and is relatively independent of the remaining code.

-Wikipedia

Functions



```php
<?php

$someText = 'The sky is <strong>blue</strong>, yes it is.';
$strippedText = strip_tags($someText);
echo $strippedText; // The sky is blue, yes it is.
```

String Functions

- Length: strlen()
- Sequence of chars: substr()
- Searching: strstr()
- Position of sequence: strpos() strrpos()
- Replacing text: str_replace()
- Case: strtolower() strtoupper() ucfirst() lcfirst() ucwords()
- Formatting: printf() sprintf() trim() ltrim() rtrim() number_format()

Length: strlen()

```
int strlen ( string $string )

$name = “Henry”;
echo strlen($name); // displays 5
echo strlen(“text”); // displays 4
```

Sequence of chars: substr()

```
string substr ( string $string , int $start [, int $length ] )

$aString = “Hello, class!”;
echo substr($aString, 0, 5); // displays ‘Hello’
echo substr($aString, 7); // displays ‘class!’
echo substr($aString, -1); // displays ‘!’
echo substr($aString, -6, -1); // displays ‘class’
```

Searching: strstr()

```
string strstr ( string $haystack , mixed $needle [, bool $before_needle = false ] )
```

```
$aString = “Hello, class!”;
echo strstr($aString, ‘cla’); // displays ‘class!’
echo strstr($aString, ‘, cl’, true); // displays ‘Hello’
echo strstr($aString, ‘abc’); // displays false “ “
```

Searching: strstr()

int strpos (string $haystack , mixed $needle [, int $offset = 0])
int strrpos (string $haystack , mixed $needle [, int $offset = 0])

```
$aString = “Hello, class! How goes it?”;
echo strpos($aString, ‘cla’); // displays 7
echo strpos($aString, ‘abc’); // displays false “
echo strrpos($aString, ‘lo’); // displays 3
// Using an offset
echo strpos($aString, ‘l’, 0); // displays 2
echo strpos($aString, ‘l’, 3); // displays 3
echo strpos($aString, ‘l’, 4); // displays 8
```

Replacing: str_replace()

```
mixed str_replace ( mixed $needle , mixed $replacement , mixed $haystack [, int &$count ] )
```

```
$aString = “Hello, class!”;
echo str_replace(‘class’, ‘Dave’, $aString); // “Hello, Dave!”
echo str_replace(‘s’, ‘$’, $aString); // “Hello, cla$$!”
echo str_replace(‘s’, ‘$’, $aString, 1); // “Hello, cla$s!”
```

Upper- and Lowercase

$text = ‘Hello, world.’;

- strtolower($text); // ‘hello, world.’
- strtoupper($text); // ‘HELLO, WORLD.’
- ucfirst($text); // ‘Hello, world.’
- lcfirst($text); // ‘hello, world.’
- ucwords($text); // ‘Hello, World.’

Case Sensitivity

Function	Case-Insensitive Equivalent
strstr()	stristr()
strpos	stripos()
strrpos()	strripos()
str_replace()	str_ireplace()

Trimming: trim(), ltrim(), rtrim()

```
string trim ( string $str [, string $charlist ] )
string ltrim ( string $str [, string $charlist ] )
string rtrim ( string $str [, string $charlist ] )

$aString = “   Hello, class!   ”;
echo trim($aString); // “Hello, class!”

$aString = “Hello, class!”;
echo trim($aString, ‘H!’); // “ello, class!”

$path = “/var/www/”;
echo ltrim($path, ‘var/’); // “/www/”
echo rtrim($path, ‘/’); // “/var/www”
```

printf() and sprintf()

- Format strings to be more *human readable*.
- General Purpose Formatting
- Uses *Conversion Specifications* as placeholders
- Each conversion specification requires an additional argument to the function
- Many *type specifiers* exist for string formatting

printf() and sprintf()

- padding output
- specifying number precision
- swapping arguments

Conversion Specifiers

- Type specifier %d   %s   %f
- Sign specifier %+d
- Padding specifier %'.5d
- Width specifier %5d   %05d
- Alignment specifier %-5s
- Precision specifier %.5f

To The Editor!

Lab Assignment

http://davehauenstein.com/nyu/INFO1-CE9224-2012-Summer/labs/class2.pdf