PHP 基本程式設計 II

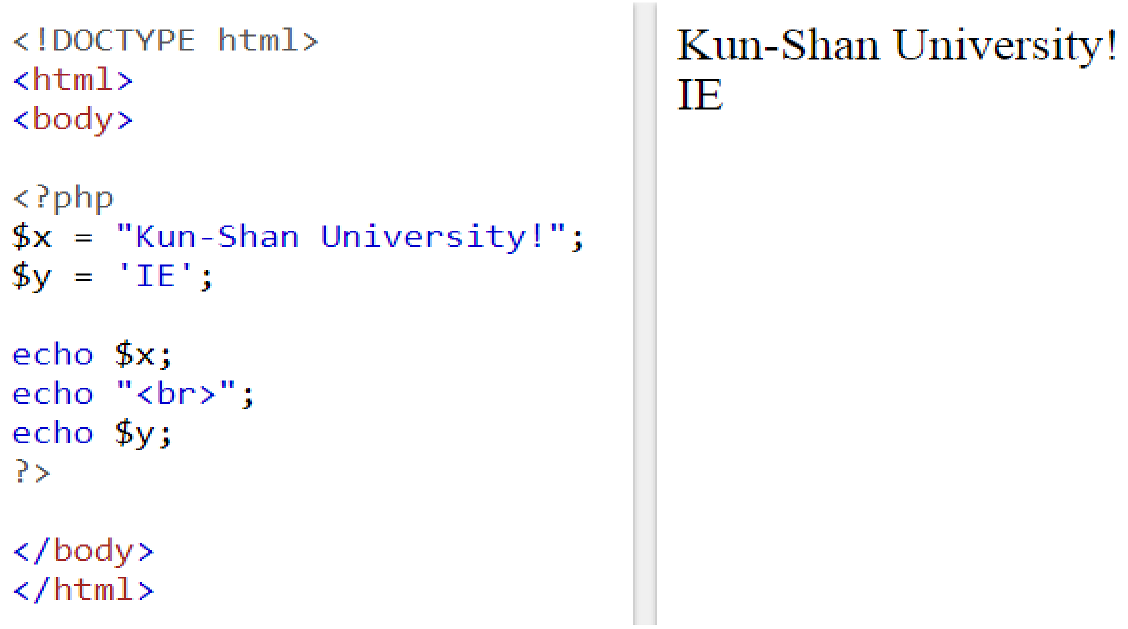
PHP Data Types

PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:
- Example

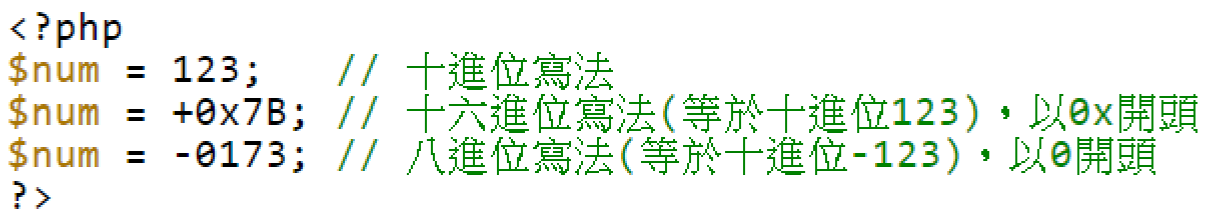


```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Kun-Shan University!";
$y = 'IE';

echo $x;
echo "<br>";
echo $y;
?>

</body>
</html>
```

```
Kun-Shan University!
IE
```

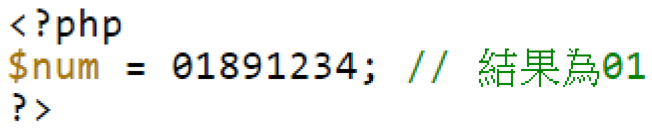
整數(Integer)

- 整數型存整數，以數學表示的話為Z = {..., -2, -1, 0, 1, 2, ...}，整數有三種宣告方式：
  - 十進位制。
  - 十六進位制，以0x開頭。
  - 八進位制，以0開頭。



```
<?php
$num = 123;   // 十進位寫法
$num = +0x7B; // 十六進位寫法(等於十進位123)，以0x開頭
$num = -0173; // 八進位寫法(等於十進位-123)，以0開頭
?>
```

  - 八進位中出現無效的數字8或9時，會忽略該數字之後的部份



```
<?php
$num = 01891234; // 結果為01
?>
```

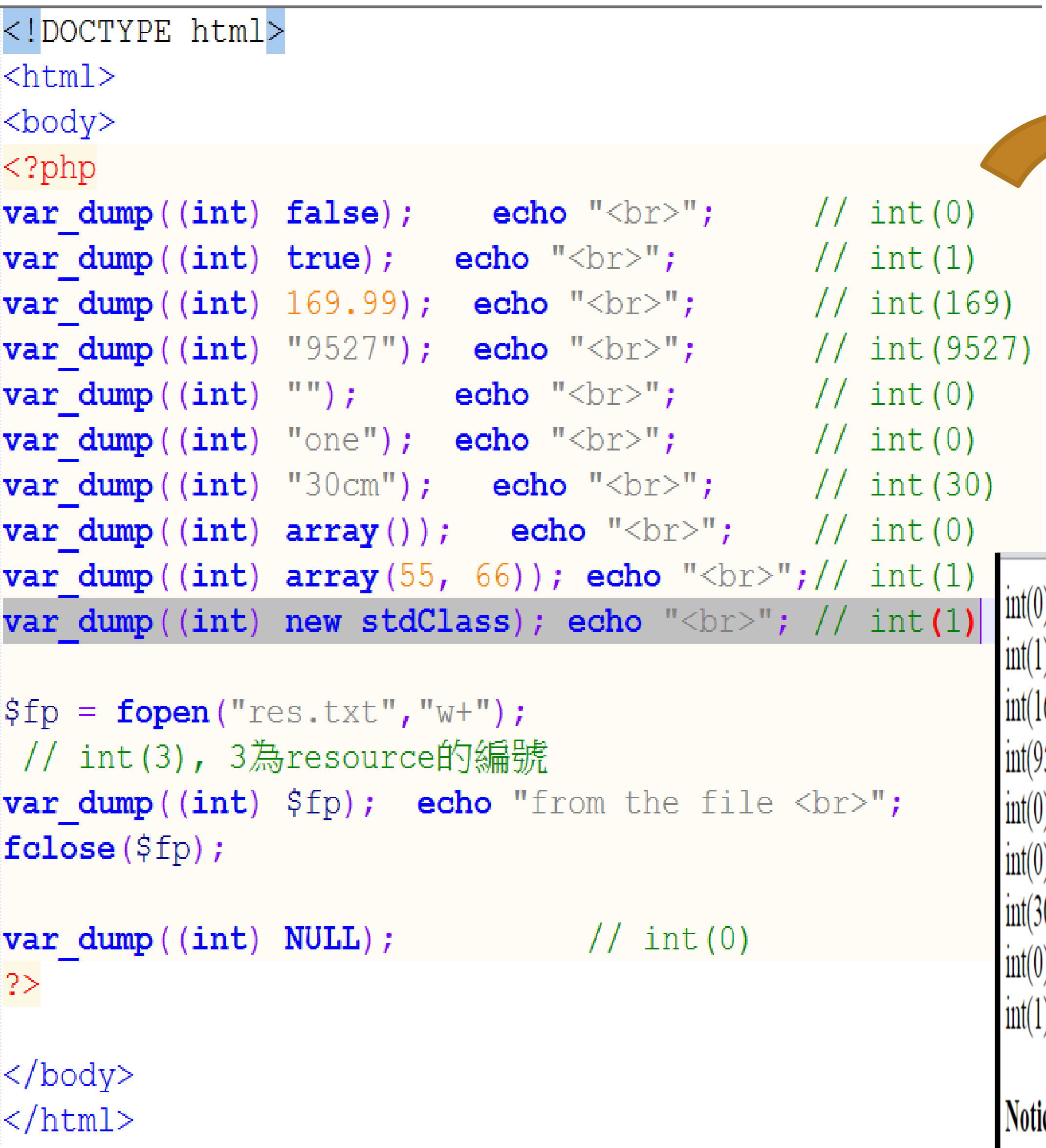
數值範圍與溢位

- 整數的數值範圍會與使用平台有關，例如32位元的平台整數使用4 byte存放，可使用PHP常數PHP_INT_SIZE取得該平台的int佔用的大小，另外可用PHP_INT_MAX取得整數的最大值。

```php
<?php
$num = PHP_INT_MAX;
var_dump($num);  // int(2147483647)
$num++;
var_dump($num);  // float(2147483648)
var_dump(28/7);  // int(4)
var_dump(25/7);  // float(3.5714285714286)
?>
```

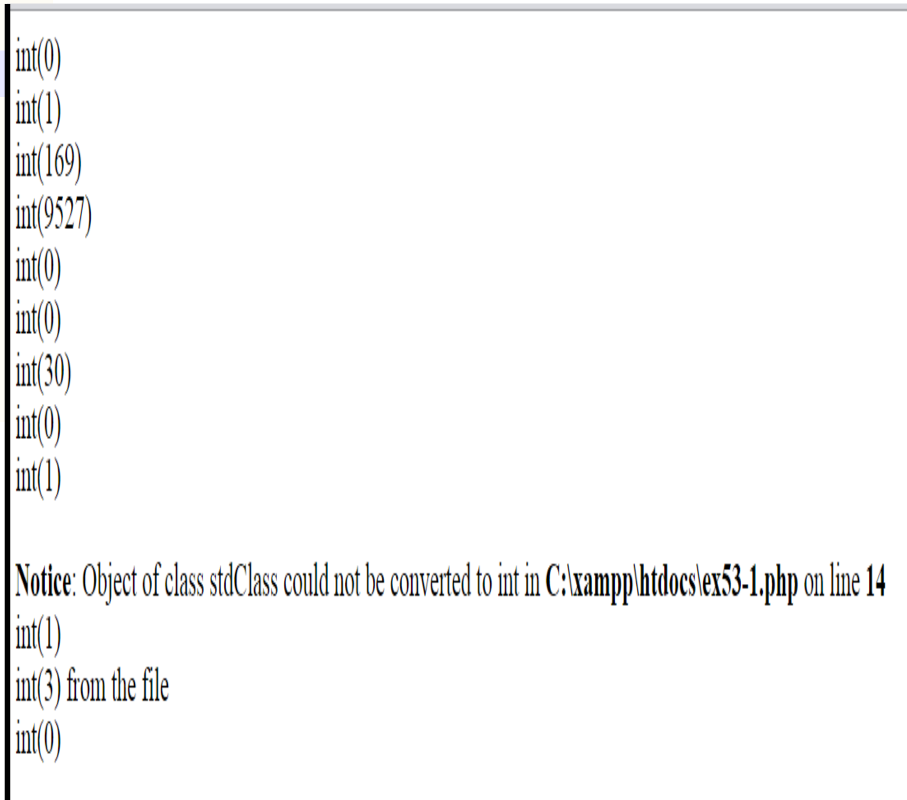
轉型

- 資料型別要轉型為整數可利用以下方式
  - 利用(int)或(integer)的強制轉型。
  - 利用intval()函式轉型。
  - 利用settype()傳入引數"int"或"integer"轉型。("int"為PHP 4.2.0之後新增)



```
<!DOCTYPE html>
<html>
<body>
<?php
var_dump((int) false);     echo "<br>";      // int(0)
var_dump((int) true);    echo "<br>";        // int(1)
var_dump((int) 169.99);   echo "<br>";       // int(169)
var_dump((int) "9527");   echo "<br>";       // int(9527)
var_dump((int) "");      echo "<br>";        // int(0)
var_dump((int) "one");   echo "<br>";        // int(0)
var_dump((int) "30cm");    echo "<br>";      // int(30)
var_dump((int) array());    echo "<br>";     // int(0)
var_dump((int) array(55, 66)); echo "<br>";// int(1)
var_dump((int) new stdClass); echo "<br>"; // int(1)

$fp = fopen("res.txt","w+");
 // int(3), 3為resource的編號
var_dump((int) $fp);  echo "from the file <br>";
fclose($fp);

var_dump((int) NULL);            // int(0)
?>

</body>
</html>
```

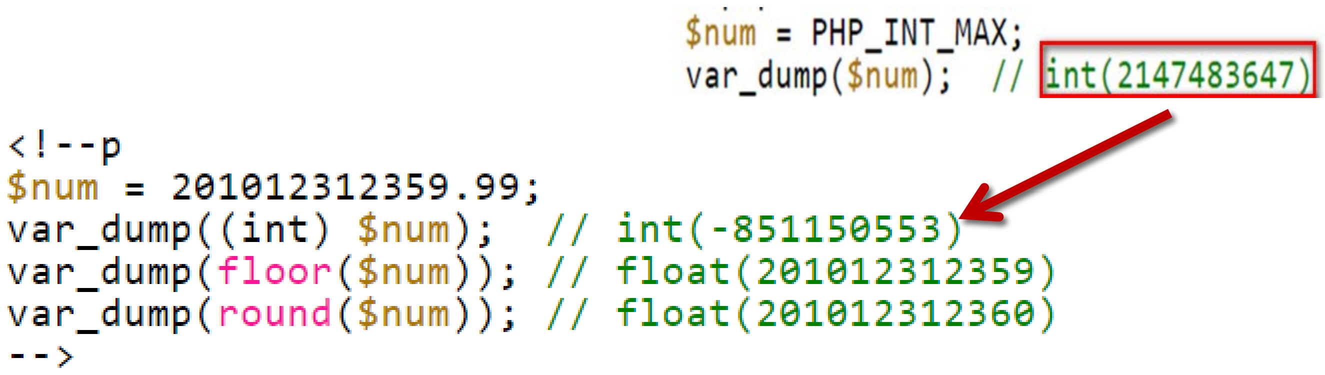
ex53-1.php



```
int(0)
int(1)
int(169)
int(9527)
int(0)
int(0)
int(30)
int(0)
int(1)

Notice: Object of class stdClass could not be converted to int in C:\xampp\htdocs\ex53-1.php on line 14
int(1)
int(3) from the file
int(0)
```

轉型

- 轉型為整數時，若有小數點則無條件捨去，但是注意，若該數字超出整數範圍時，會出現不如預期的結果，如果你是想做『取整數』的動作的話，可以使用floor()函式來處理。另外，若要四捨五入的話可使用round()函式

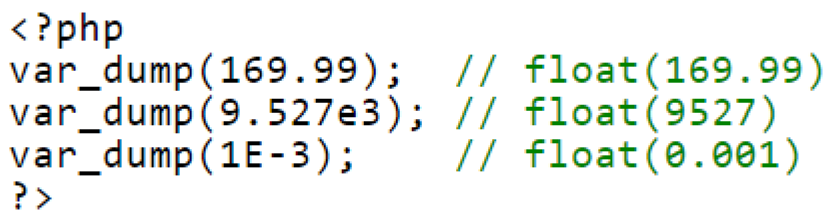


```
$num = PHP_INT_MAX;
var_dump($num);  // int(2147483647)
```

```
<!--p
$num = 201012312359.99;
var_dump((int) $num);   // int(-851150553)
var_dump(floor($num)); // float(201012312359)
var_dump(round($num)); // float(201012312360)
-->
```

浮點數(Float)

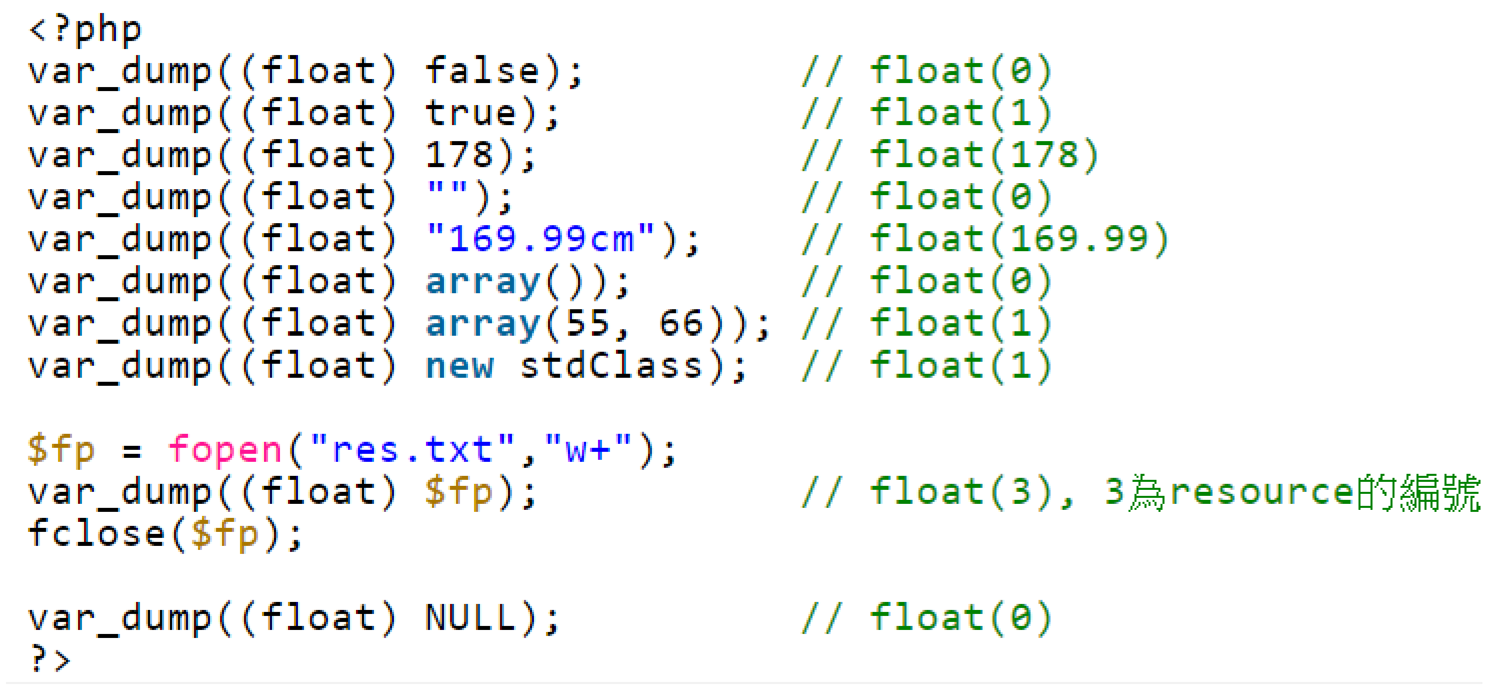
- 浮點數英文Floating point number簡稱float或double，有兩種寫法：
  - 一般數字。
  - 科學記號表示法，使用e表示10的幾次方，e不分大小寫。



```
<?php
var_dump(169.99);   // float(169.99)
var_dump(9.527e3); // float(9527)
var_dump(1E-3);     // float(0.001)
?>
```

轉型

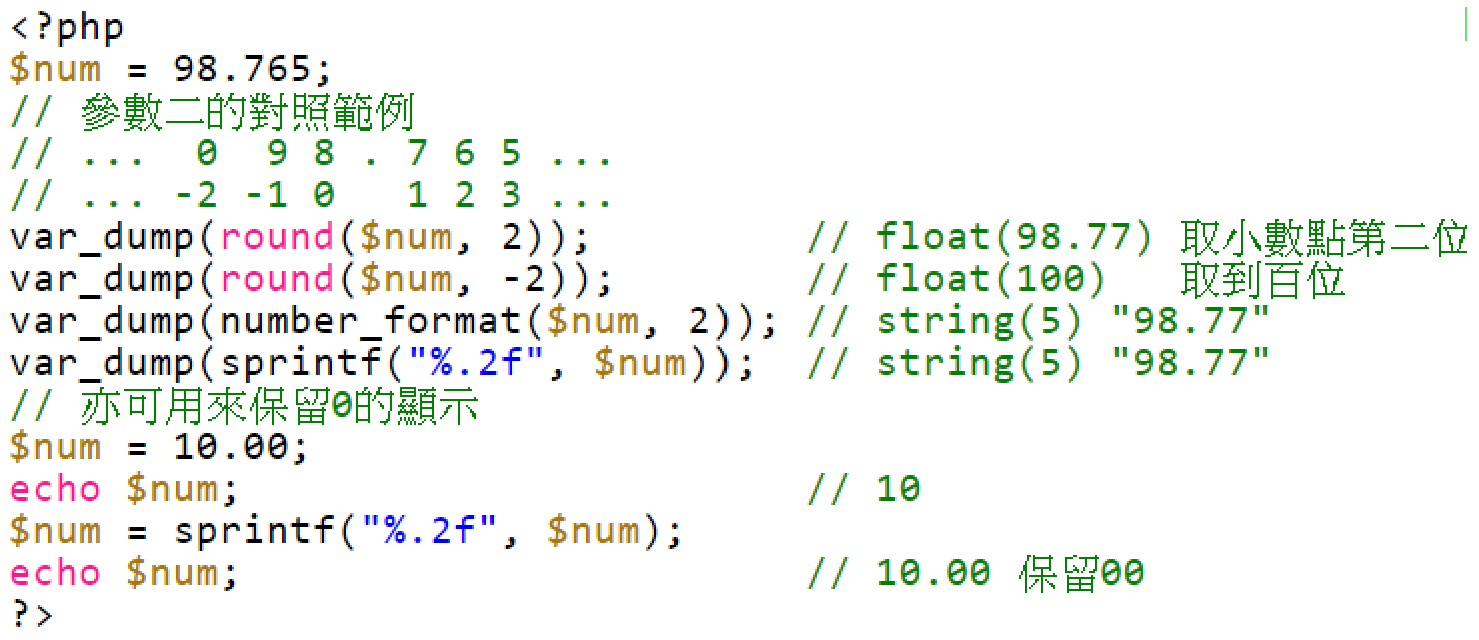
- 利用(float)或(double)的強制轉型。
- 利用floatval()或doubleval()函式轉型。(floatval()為PHP 4.2.0之後新增)
- 利用settype()傳入引數"float"或"double"轉型。("float"為PHP 4.2.0之後新增)



```php
<?php
var_dump((float) false);          // float(0)
var_dump((float) true);           // float(1)
var_dump((float) 178);            // float(178)
var_dump((float) "");             // float(0)
var_dump((float) "169.99cm");     // float(169.99)
var_dump((float) array());        // float(0)
var_dump((float) array(55, 66));  // float(1)
var_dump((float) new stdClass);   // float(1)

$fp = fopen("res.txt","w+");
var_dump((float) $fp);            // float(3), 3為resource的編號
fclose($fp);

var_dump((float) NULL);           // float(0)
?>
```

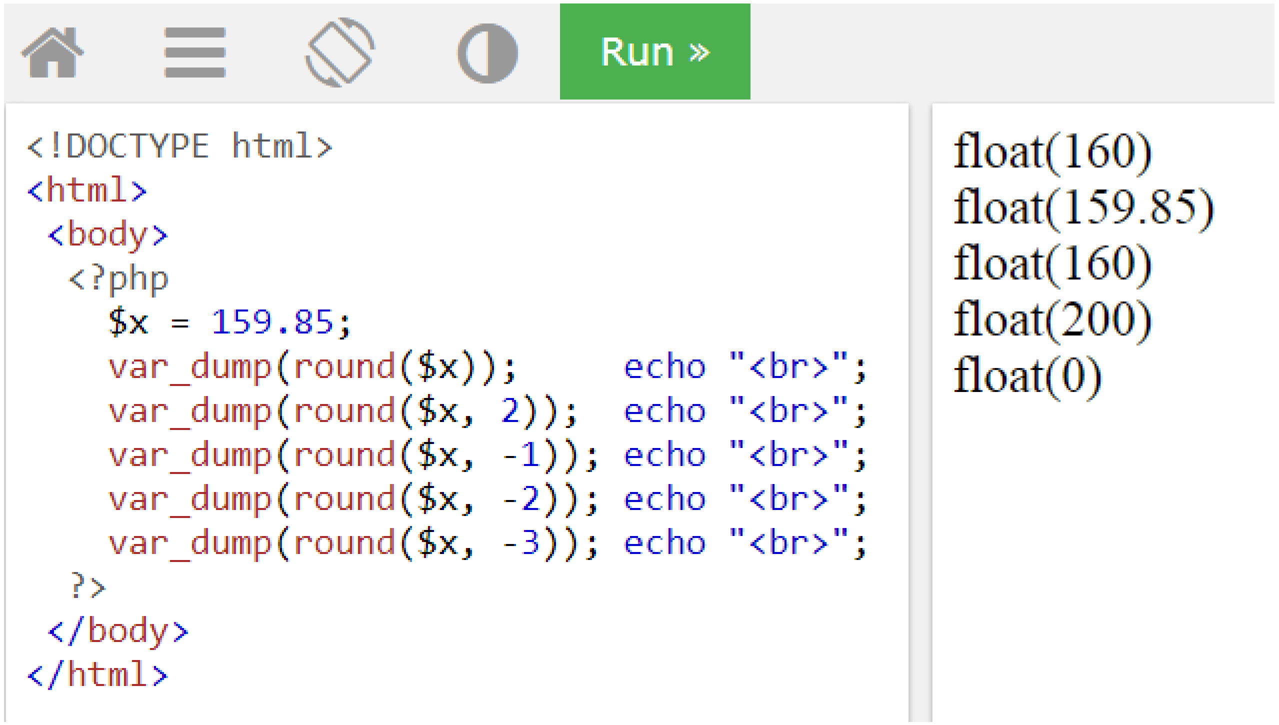
取小數點位數

- 如果我們想要只顯示到小數點第二位，可以用幾種方式處理：
  - 用round()函式，前面也曾使用過，只要再進一步指定第二個參數，就可以拿來取小數點位數。
  - 使用number_format()函式，第二個參數表示到小數點第幾位。
  - 使用sprintf()函數，%f表示輸出浮點數，%.2f表示小數點第二位。



```
<?php
$num = 98.765;
// 參數二的對照範例
// ...  0  9 8 . 7 6 5 ...
// ... -2 -1 0   1 2 3 ...
var_dump(round($num, 2));          // float(98.77) 取小數點第二位
var_dump(round($num, -2));         // float(100)   取到百位
var_dump(number_format($num, 2)); // string(5) "98.77"
var_dump(sprintf("%.2f", $num));  // string(5) "98.77"
// 亦可用來保留0的顯示
$num = 10.00;
echo $num;                         // 10
$num = sprintf("%.2f", $num);
echo $num;                         // 10.00 保留00
?>
```

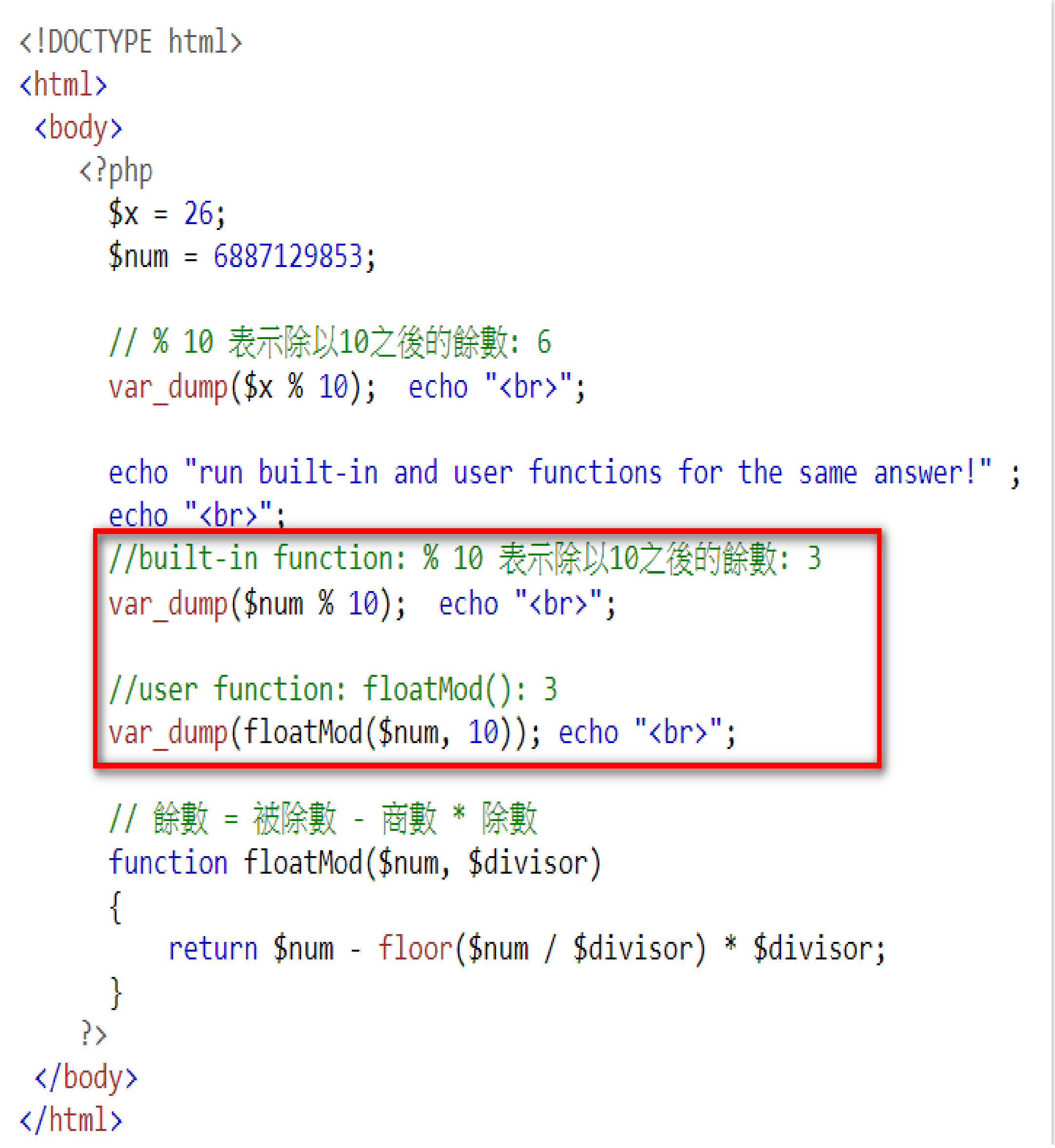
Continued



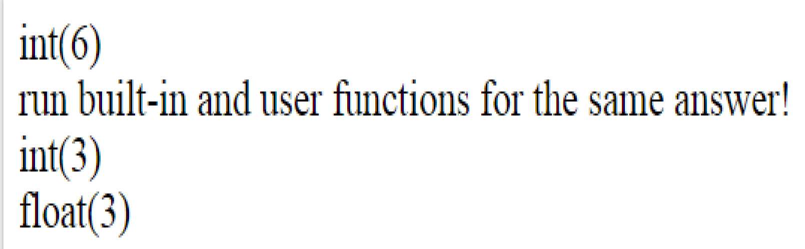
```
<!DOCTYPE html>
<html>
 <body>
  <?php
    $x = 159.85;
    var_dump(round($x));      echo "<br>";
    var_dump(round($x, 2));   echo "<br>";
    var_dump(round($x, -1));  echo "<br>";
    var_dump(round($x, -2));  echo "<br>";
    var_dump(round($x, -3));  echo "<br>";
  ?>
 </body>
</html>
```

```
float(160)
float(159.85)
float(160)
float(200)
float(0)
```

取餘數- build-in vs user function



```
<!DOCTYPE html>
<html>
 <body>
    <?php
      $x = 26;
      $num = 6887129853;

      // % 10 表示除以10之後的餘數: 6
      var_dump($x % 10);  echo "<br>";

      echo "run built-in and user functions for the same answer!" ;
      echo "<br>";
      //built-in function: % 10 表示除以10之後的餘數: 3
      var_dump($num % 10);  echo "<br>";

      //user function: floatMod(): 3
      var_dump(floatMod($num, 10)); echo "<br>";

      // 餘數 = 被除數 - 商數 * 除數
      function floatMod($num, $divisor)
      {
          return $num - floor($num / $divisor) * $divisor;
      }
    ?>
 </body>
</html>
```



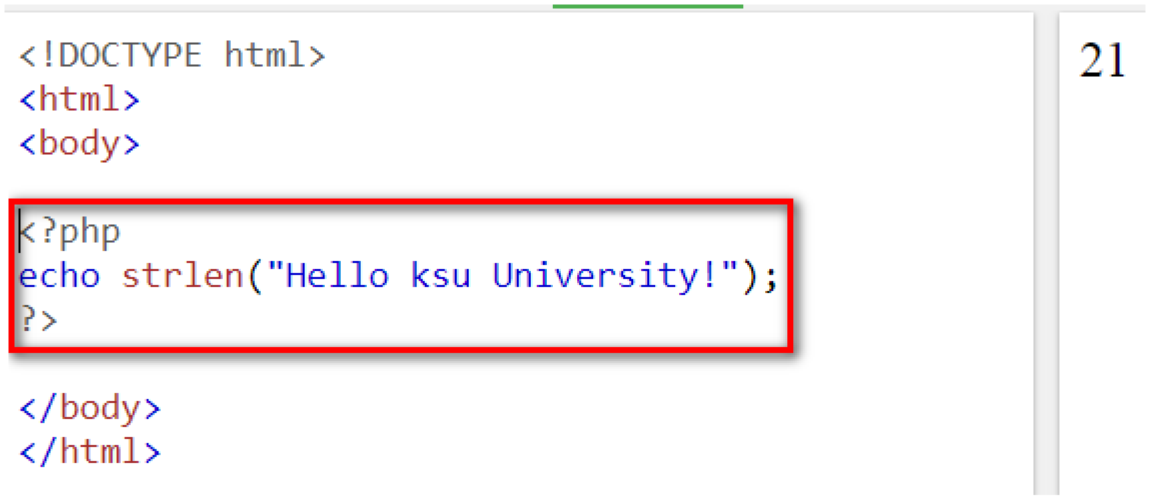
```
int(6)
run built-in and user functions for the same answer!
int(3)
float(3)
```

ex53-2.php

PHP String

strlen() - Return the Length of a String

- In this section we will look at some commonly used functions to manipulate strings.
- The PHP strlen() function returns the length of a string.

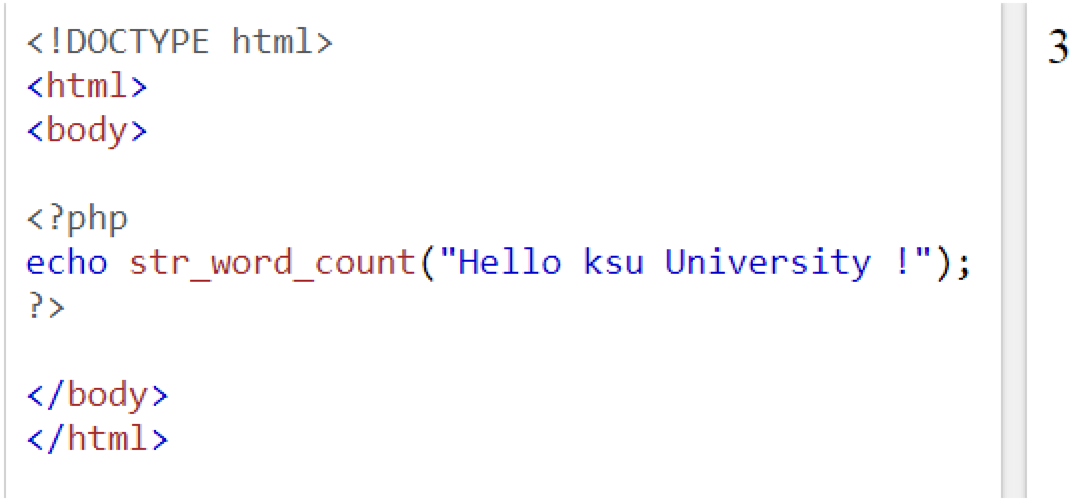


```
<!DOCTYPE html>
<html>
<body>

<?php
echo strlen("Hello ksu University!");
?>

</body>
</html>
```

```
21
```

str_word_count()

- str_word_count() - Count Words in a String

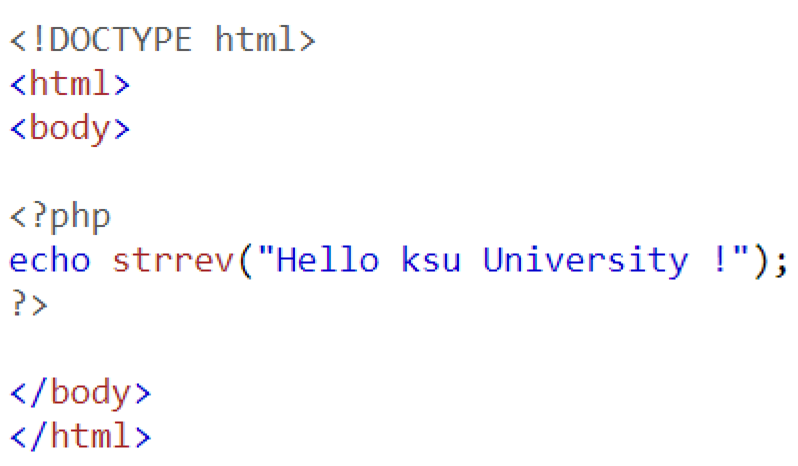


```
<!DOCTYPE html>
<html>
<body>

<?php
echo str_word_count("Hello ksu University !");
?>

</body>
</html>
```

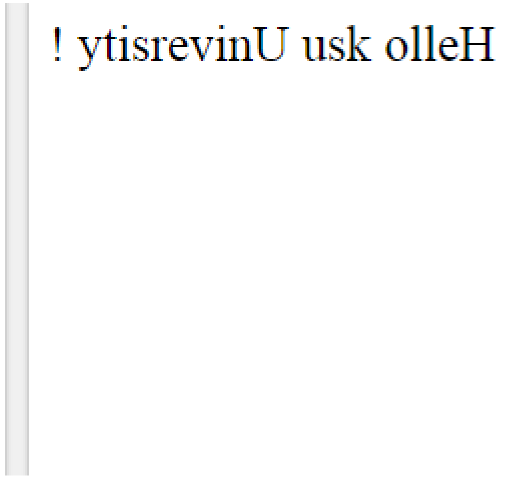
```
3
```

strrev()

- strrev() - Reverse a String



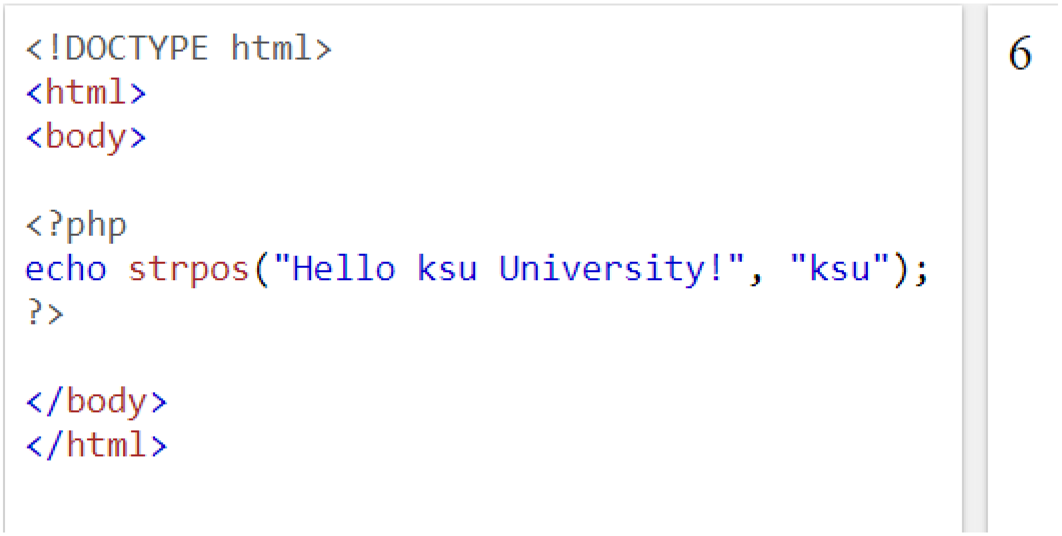
```
<!DOCTYPE html>
<html>
<body>

<?php
echo strrev("Hello ksu University !");
?>

</body>
</html>
```



```
! ytisrevinU usk olleH
```

strpos()

- strpos() - Search For a Text Within a String
- The first character position in a string is **0** (not 1).

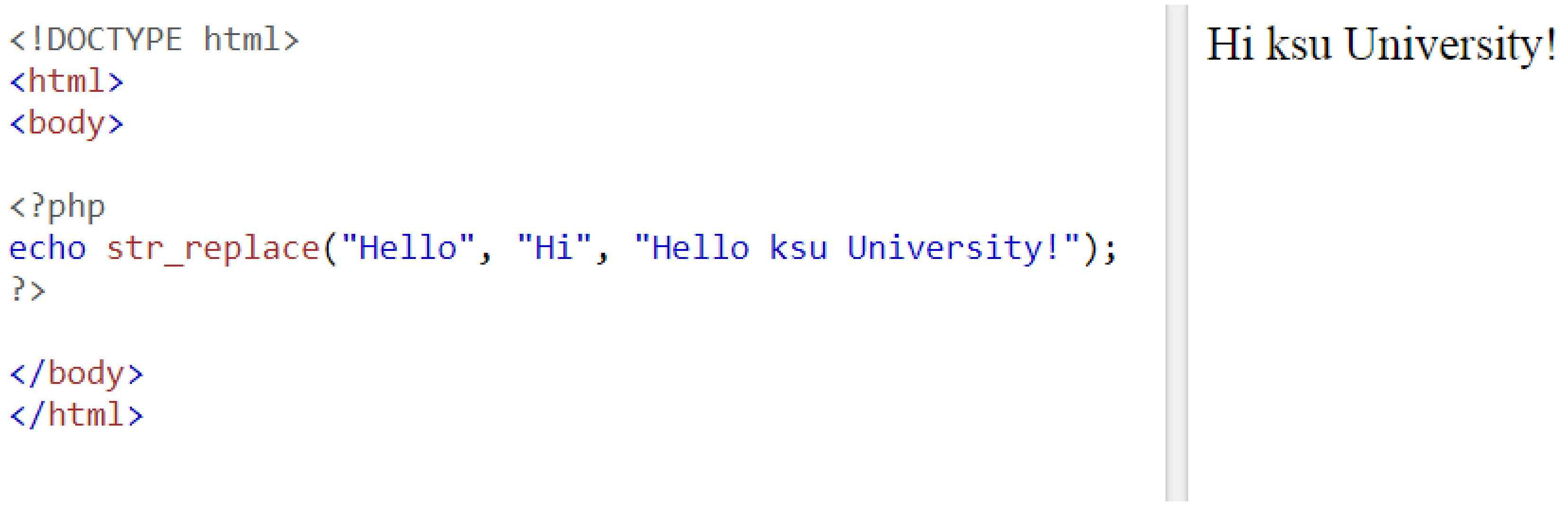


```
<!DOCTYPE html>
<html>
<body>

<?php
echo strpos("Hello ksu University!", "ksu");
?>

</body>
</html>
```

```
6
```

str_replace()

- str_replace() - Replace Text Within a String



```
<!DOCTYPE html>
<html>
<body>

<?php
echo str_replace("Hello", "Hi", "Hello ksu University!");
?>

</body>
</html>
```

```
Hi ksu University!
```

PHP Numbers

PHP Numbers

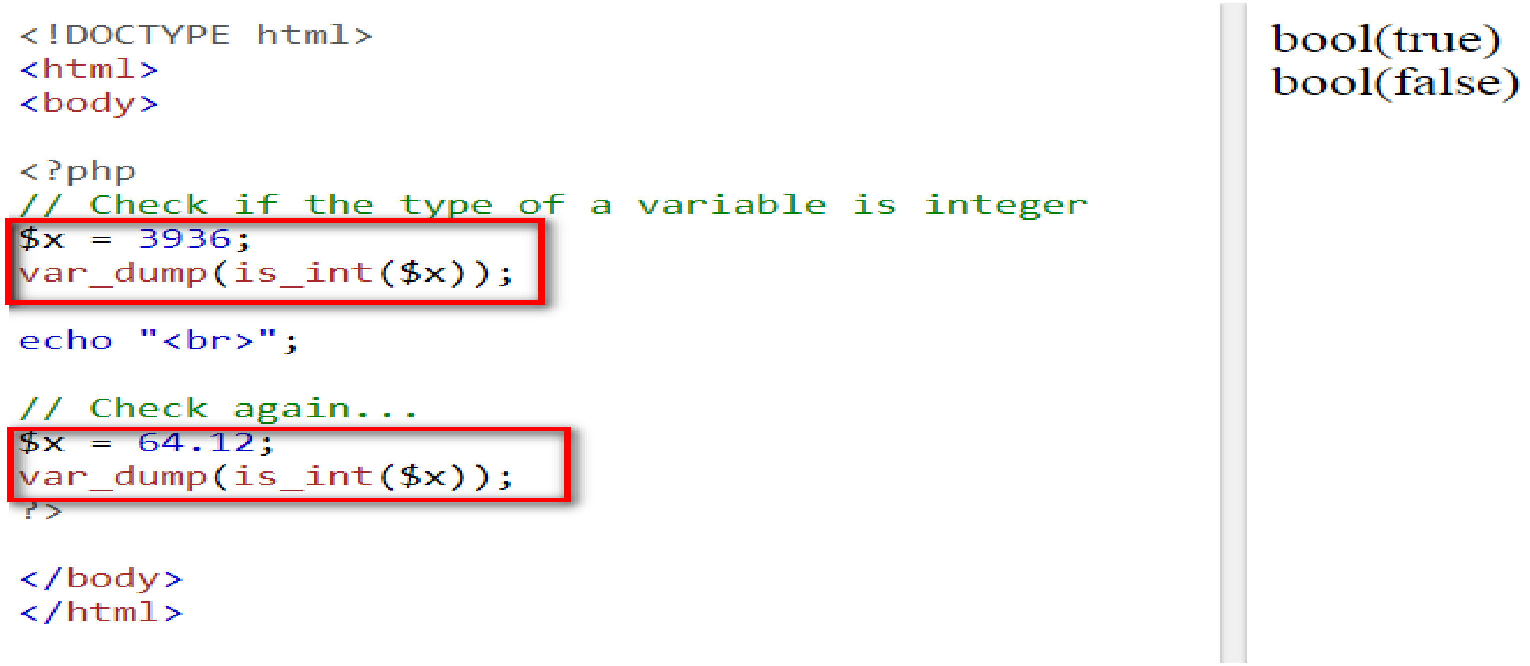
- One thing to notice about PHP is that it provides automatic data type conversion.
- So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.
- This automatic conversion can sometimes break your code.

PHP Integers

- An integer is a number without any decimal part.
- For instance, 2, 256, -256, 10358, -179567 are all integers. While 7.56, 10.0, 150.67 are floats.
- So, an integer data type is a non-decimal number between -2147483648 and 2147483647. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.
- Another important thing to know is that even if 4 * 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

PHP Integers

- PHP has the following functions to check if the type of a variable is integer:
  - is_int()
  - is_integer() - alias of is_int()
  - is_long() - alias of is_int()

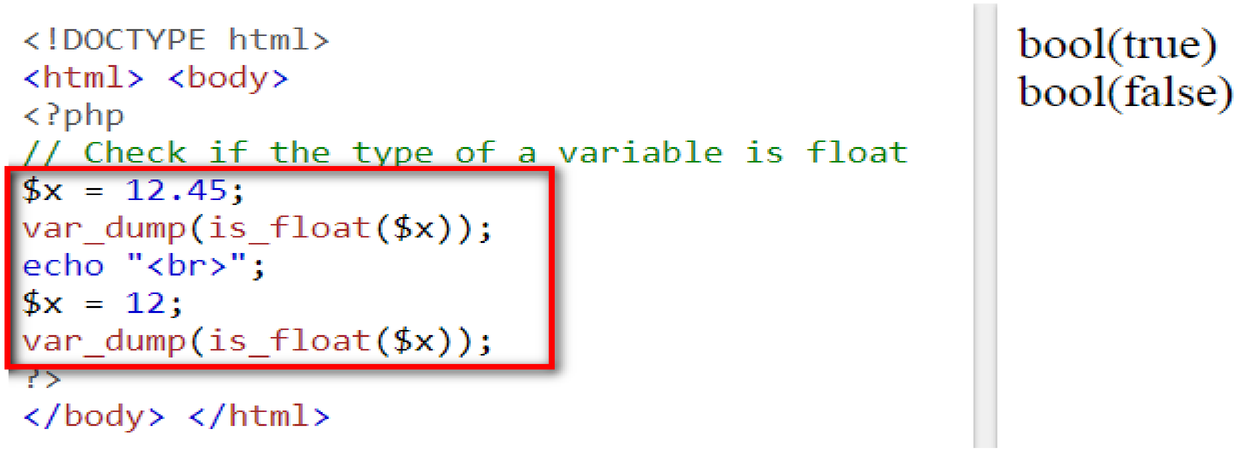


```
<!DOCTYPE html>
<html>
<body>

<?php
// Check if the type of a variable is integer
$x = 3936;
var_dump(is_int($x));

echo "<br>";

// Check again...
$x = 64.12;
var_dump(is_int($x));
?>

</body>
</html>
```

```
bool(true)
bool(false)
```

PHP Floats

- A float is a number with a decimal point or a number in exponential form.
- 2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.
- The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits. PHP has the following functions to check if the type of a variable is float:
  - is_float()
  - is_double() - alias of is_float()

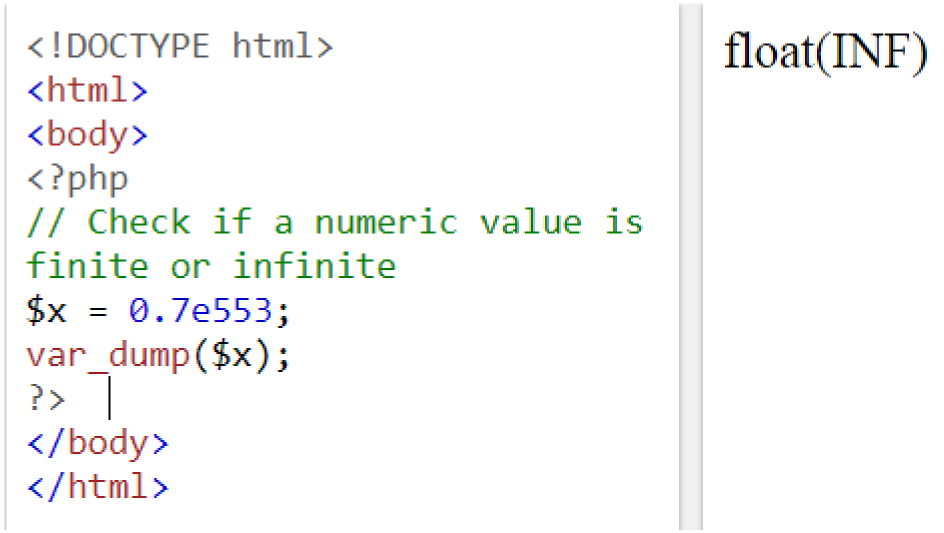


```
<!DOCTYPE html>
<html> <body>
<?php
// Check if the type of a variable is float
$x = 12.45;
var_dump(is_float($x));
echo "<br>";
$x = 12;
var_dump(is_float($x));
?>
</body> </html>
```

```
bool(true)
bool(false)
```

PHP Infinity

- A numeric value that is larger than PHP_FLOAT_MAX is considered infinite.
- PHP has the following functions to check if a numeric value is finite or infinite:
  - is_finite()
  - is_infinite()
- However, the PHP var_dump() function returns the data type and value:

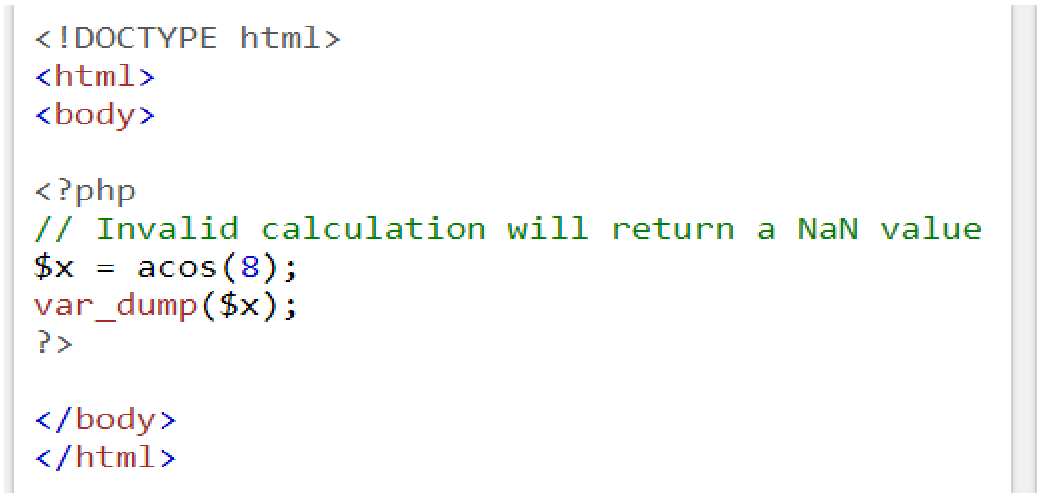


```
<!DOCTYPE html>
<html>
<body>
<?php
// Check if a numeric value is
finite or infinite
$x = 0.7e553;
var_dump($x);
?>
</body>
</html>
```

```
float(INF)
```

PHP NaN

- NaN stands for Not a Number.
- NaN is used for impossible mathematical operations.
- PHP has the following functions to check if a value is not a number:
  - is_nan()
- However, the PHP var_dump() function returns the data type and value:

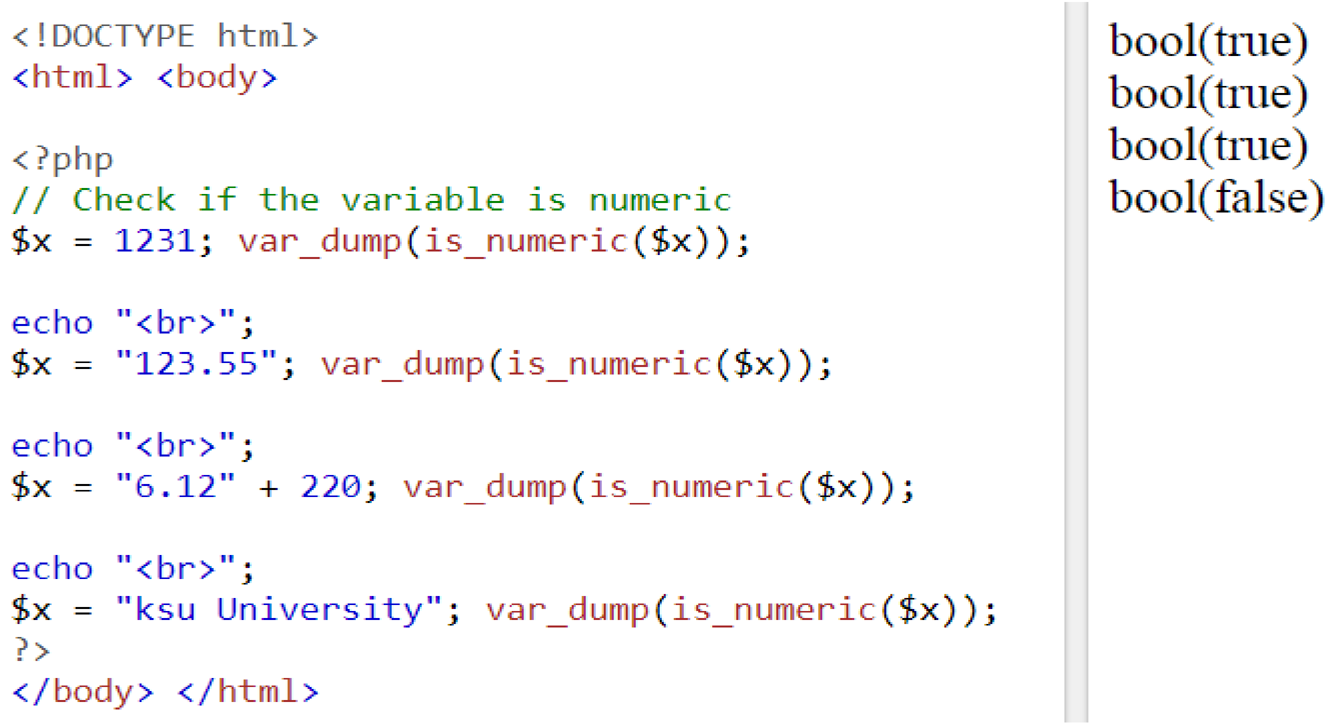


```
<!DOCTYPE html>
<html>
<body>

<?php
// Invalid calculation will return a NaN value
$x = acos(8);
var_dump($x);
?>

</body>
</html>
```

```
float(NAN)
```

PHP Numerical Strings

- The PHP is_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

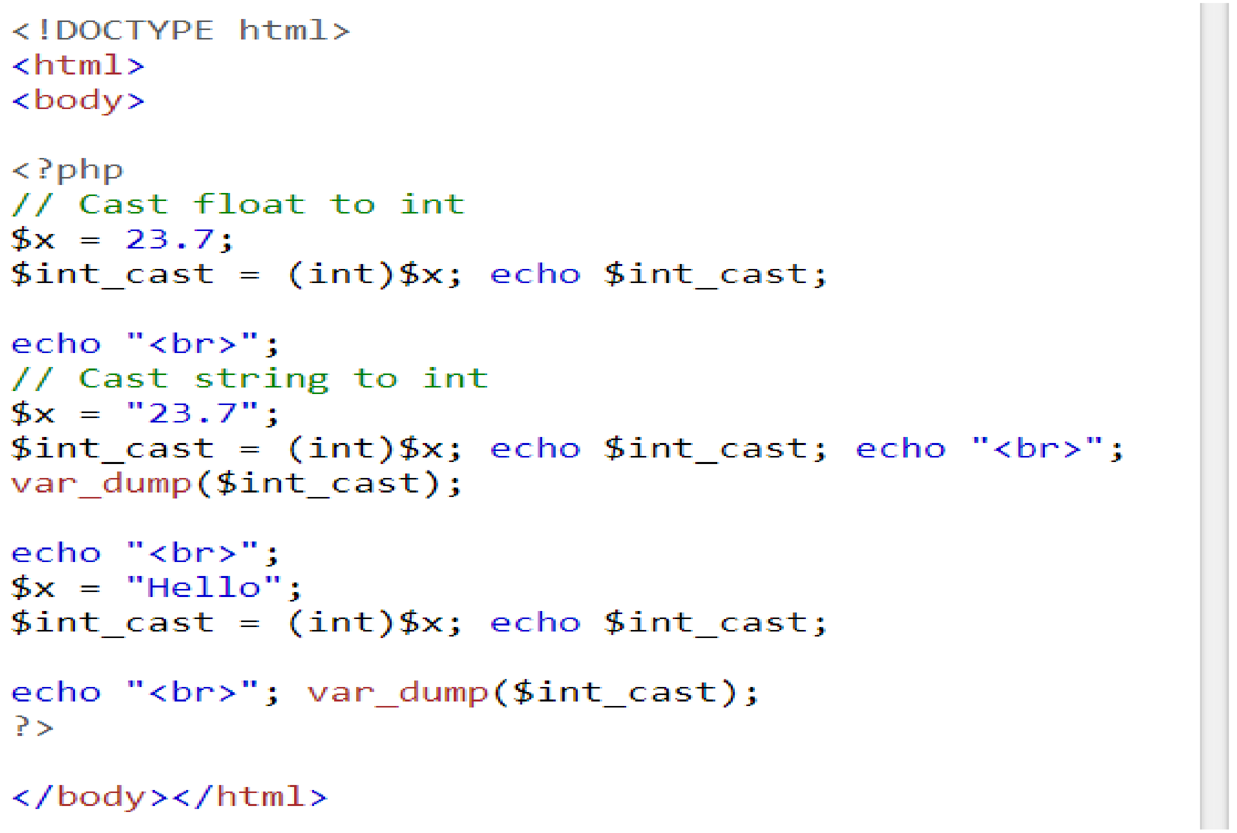


```
<!DOCTYPE html>
<html> <body>

<?php
// Check if the variable is numeric
$x = 1231; var_dump(is_numeric($x));

echo "<br>";
$x = "123.55"; var_dump(is_numeric($x));

echo "<br>";
$x = "6.12" + 220; var_dump(is_numeric($x));

echo "<br>";
$x = "ksu University"; var_dump(is_numeric($x));
?>
</body> </html>
```

```
bool(true)
bool(true)
bool(true)
bool(false)
```

PHP Casting Strings and Floats

- PHP Casting Strings and Floats to Integers
- Sometimes you need to cast a numerical value into another data type. The (int), (integer), or intval() function are often used to convert a value to an integer.



```
<!DOCTYPE html>
<html>
<body>

<?php
// Cast float to int
$x = 23.7;
$int_cast = (int)$x; echo $int_cast;

echo "<br>";
// Cast string to int
$x = "23.7";
$int_cast = (int)$x; echo $int_cast; echo "<br>";
var_dump($int_cast);

echo "<br>";
$x = "Hello";
$int_cast = (int)$x; echo $int_cast;

echo "<br>"; var_dump($int_cast);
?>

</body></html>
```

```
23
23
int(23)
0
int(0)
```

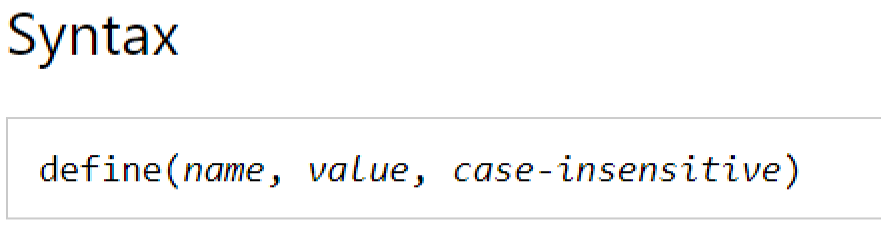
PHP Constants

PHP Constants

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- **Note:** Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

- To create a constant, use the define() function.
- Parameters:
  - *name*: Specifies the name of the constant
  - *value*: Specifies the value of the constant
  - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

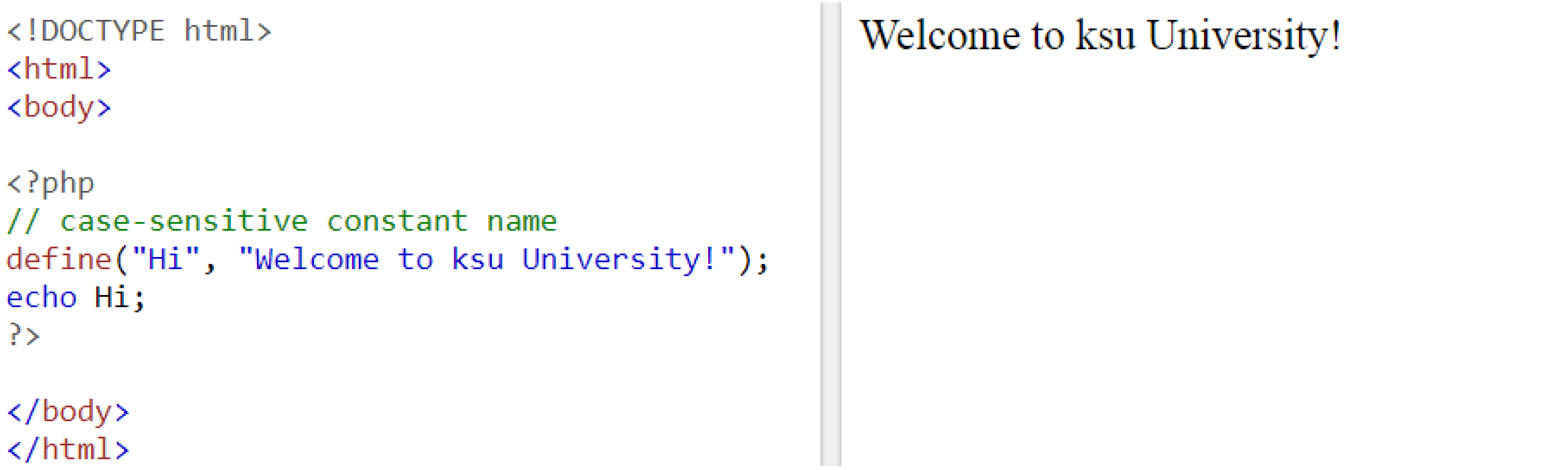


Syntax

```
define(name, value, case-insensitive)
```

Create a PHP Constant

- To create a constant, use the define() function without the 3rd parameter. Create a constant with a **case-sensitive** name:

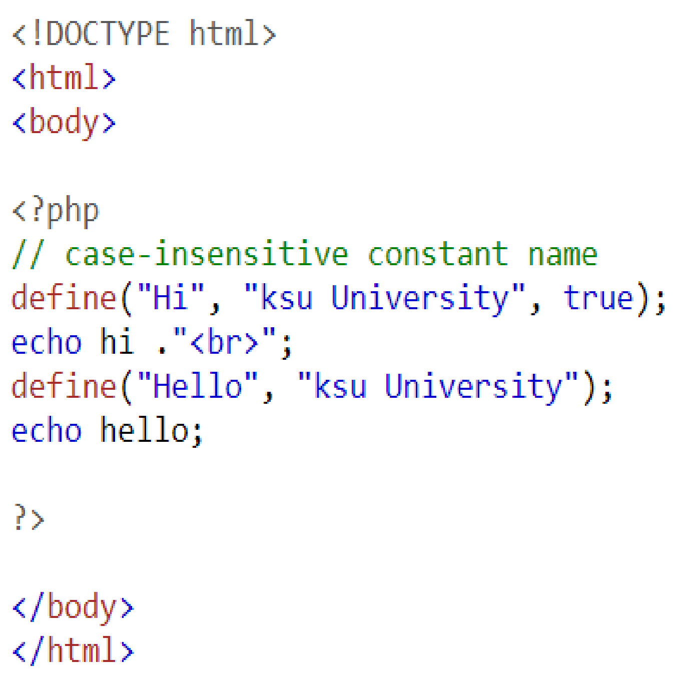


```
<!DOCTYPE html>
<html>
<body>

<?php
// case-sensitive constant name
define("Hi", "Welcome to ksu University!");
echo Hi;
?>

</body>
</html>
```

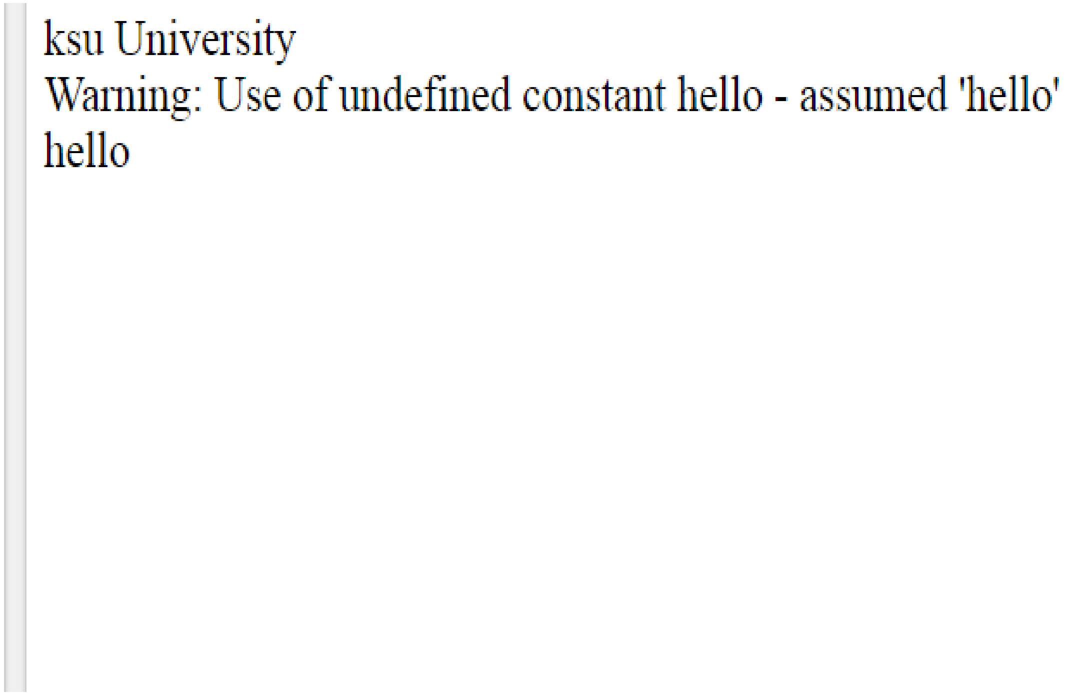
```
Welcome to ksu University!
```

Create a PHP Constant

- Create a constant with a **case-insensitive** name:



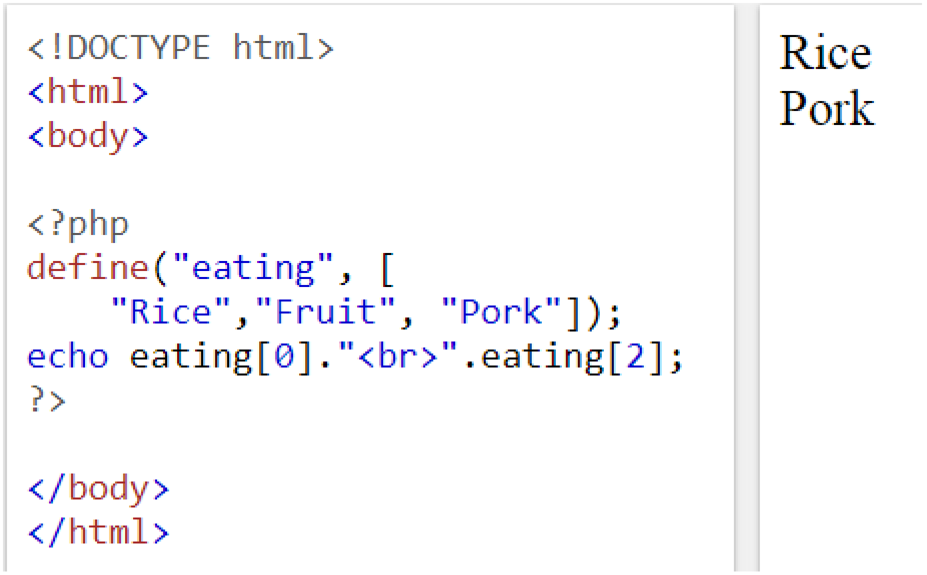
```
<!DOCTYPE html>
<html>
<body>

<?php
// case-insensitive constant name
define("Hi", "ksu University", true);
echo hi ."<br>";
define("Hello", "ksu University");
echo hello;

?>

</body>
</html>
```



```
ksu University
Warning: Use of undefined constant hello - assumed 'hello'
hello
```

PHP Constant Arrays

- In PHP7, you can create an Array constant using the define() function.

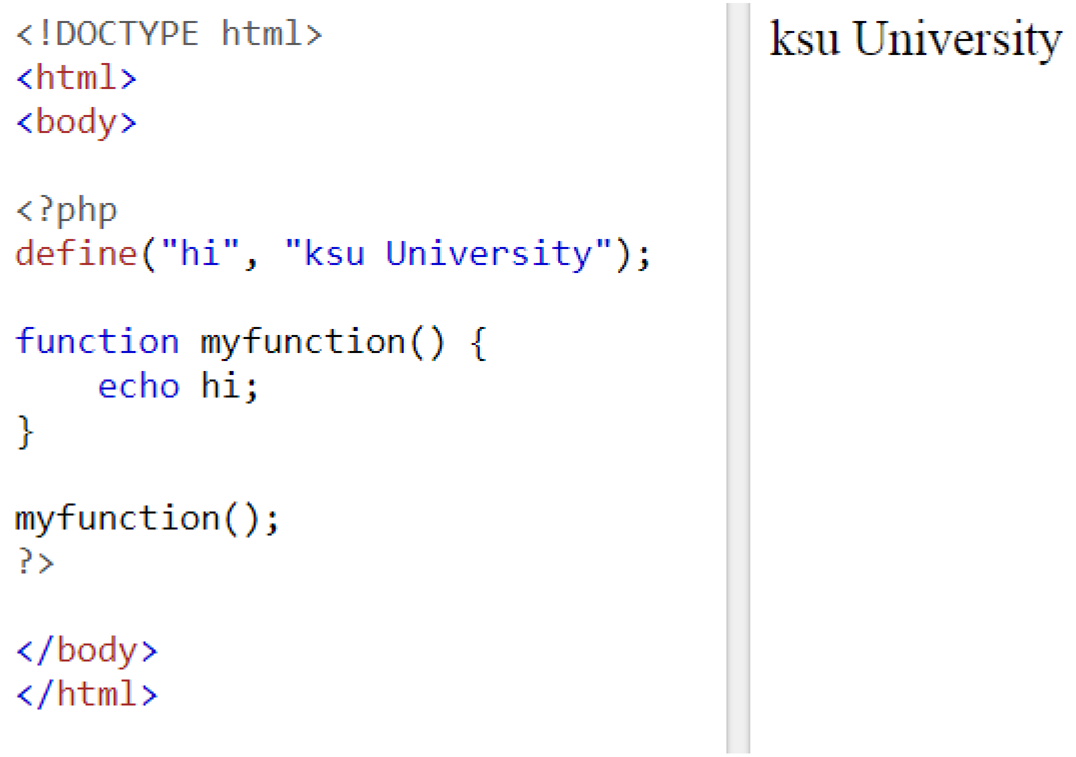


```
<!DOCTYPE html>
<html>
<body>

<?php
define("eating", [
    "Rice","Fruit", "Pork"]);
echo eating[0]."<br>".eating[2];
?>

</body>
</html>
```

```
Rice
Pork
```

Constants are Global

- Constants are automatically global and can be used across the entire script.



```
<!DOCTYPE html>
<html>
<body>

<?php
define("hi", "ksu University");

function myfunction() {
    echo hi;
}

myfunction();
?>

</body>
</html>
```

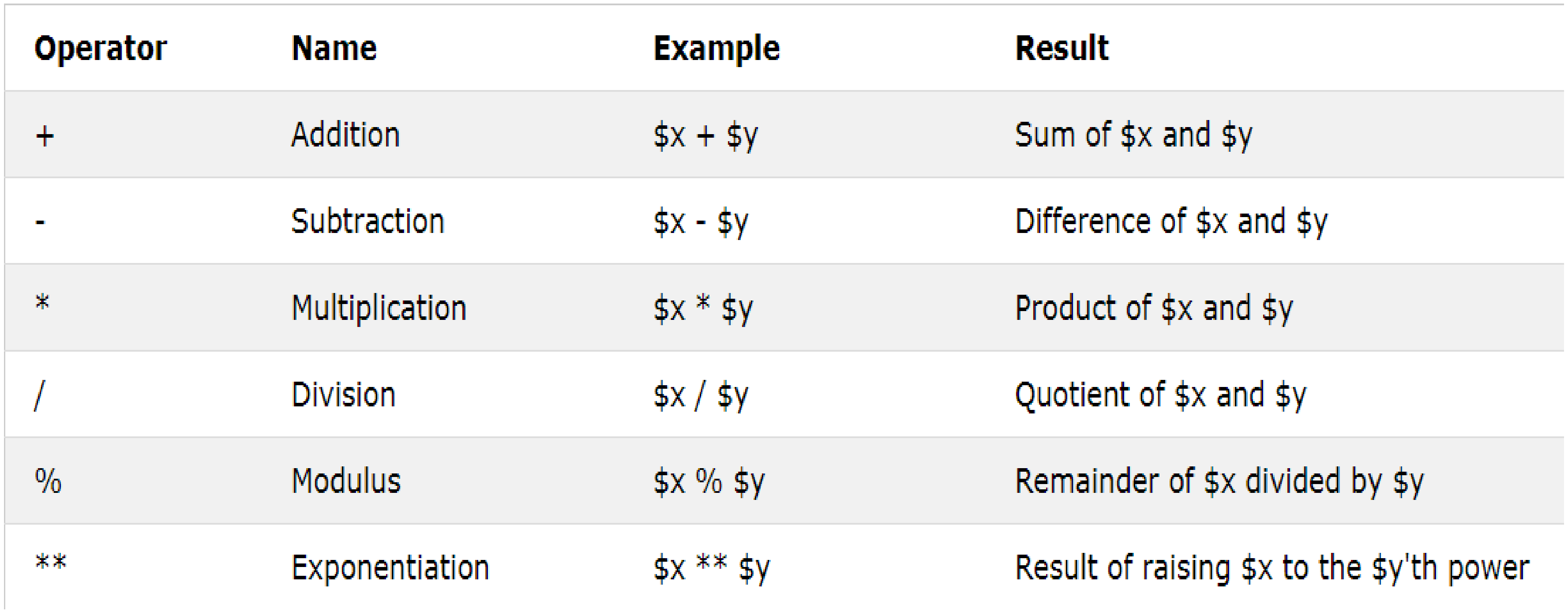
```
ksu University
```

PHP Operators

PHP Operators

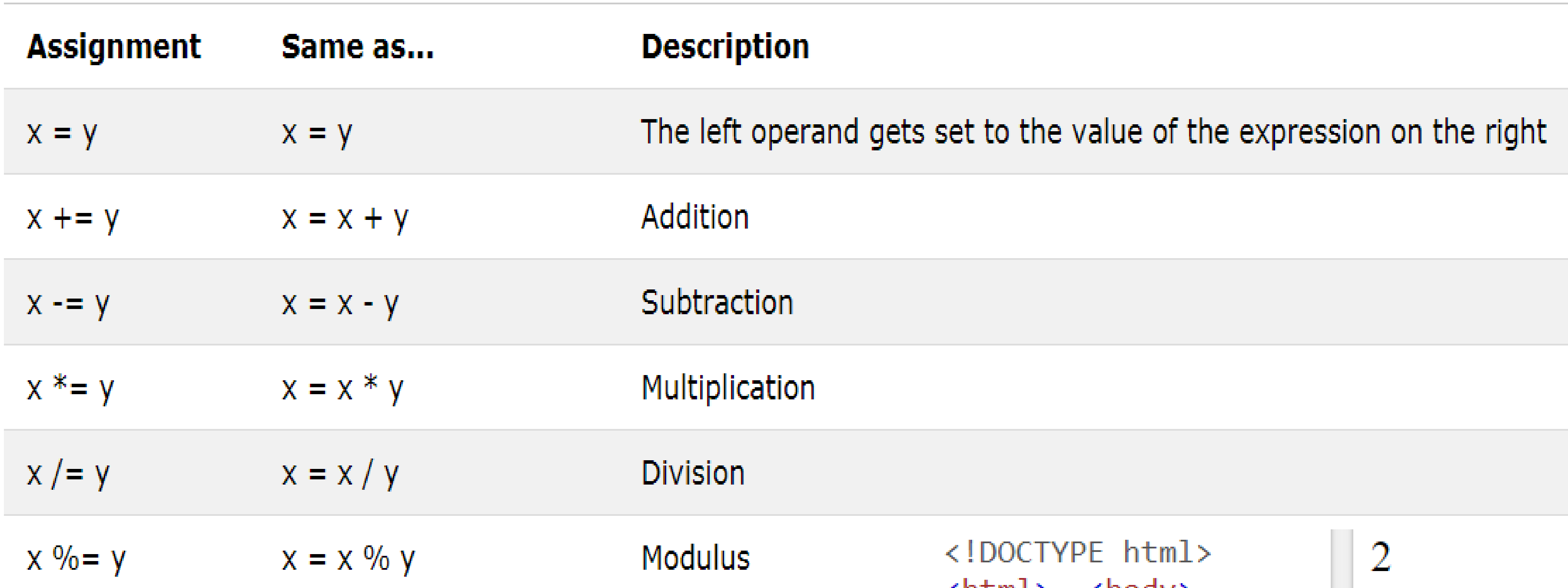
- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators
  - Conditional assignment operators

PHP Arithmetic Operators

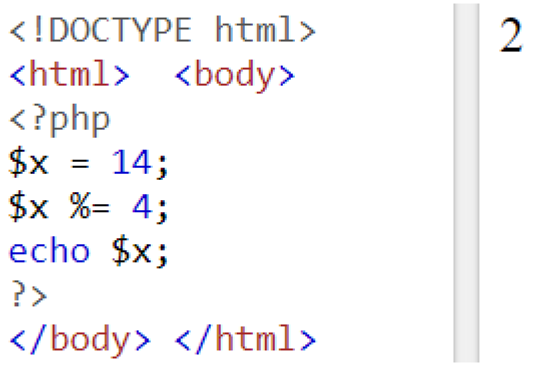


Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators



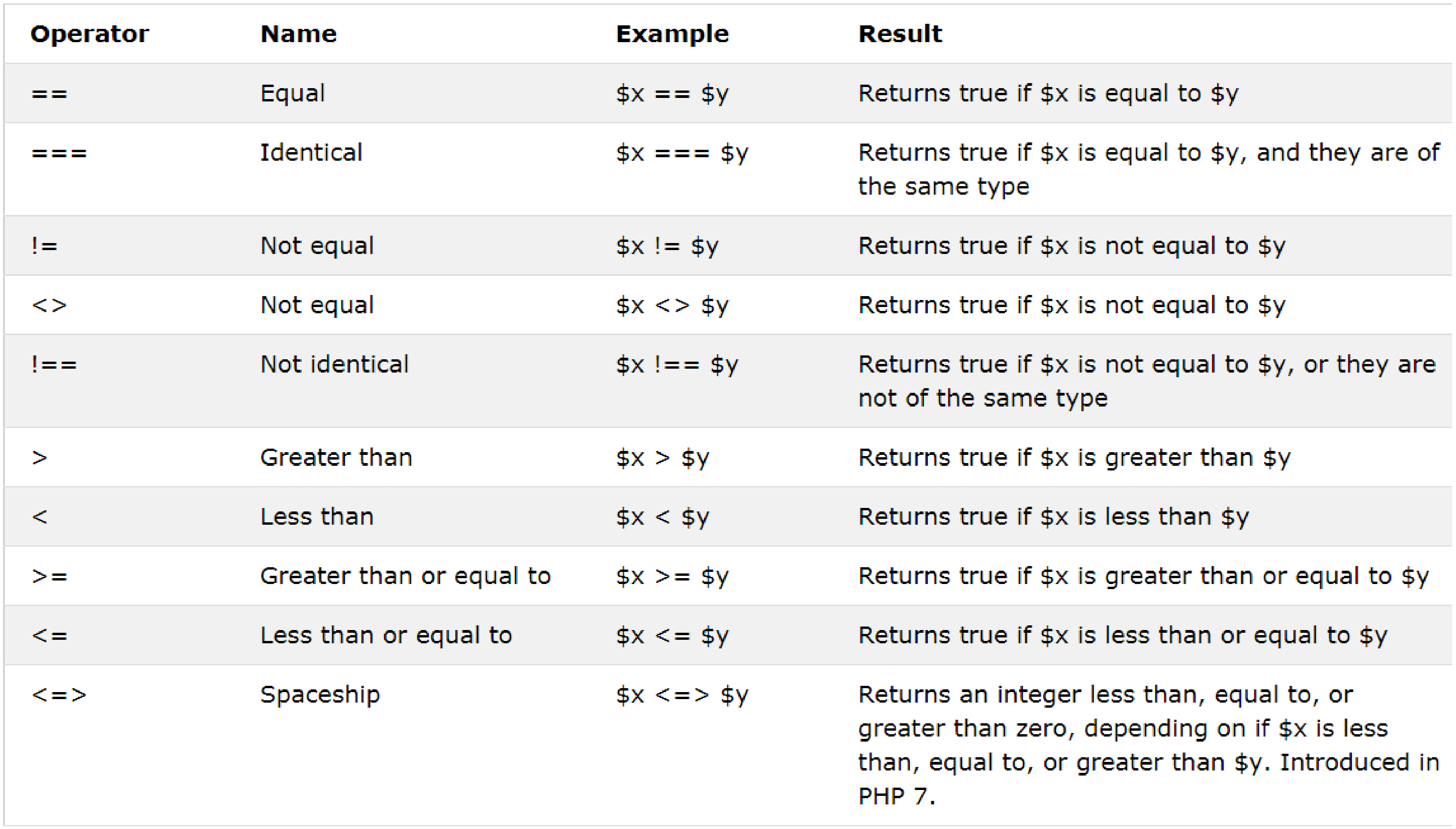
Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus



```
<!DOCTYPE html>
<html>  <body>
<?php
$x = 14;
$x %= 4;
echo $x;
?>
</body> </html>
```

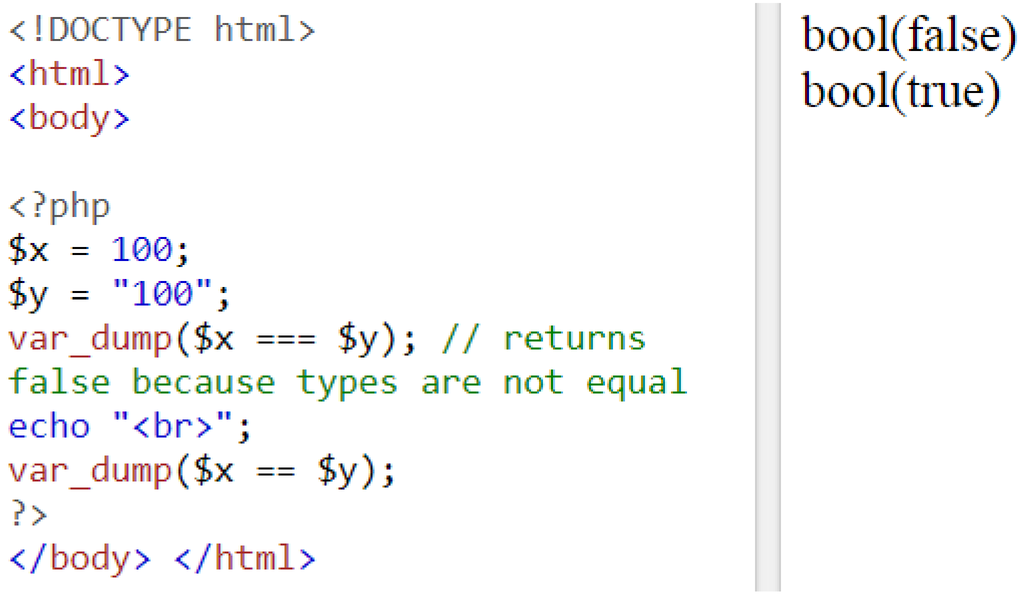
```
2
```

PHP Comparison Operators



Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Comparison Operators

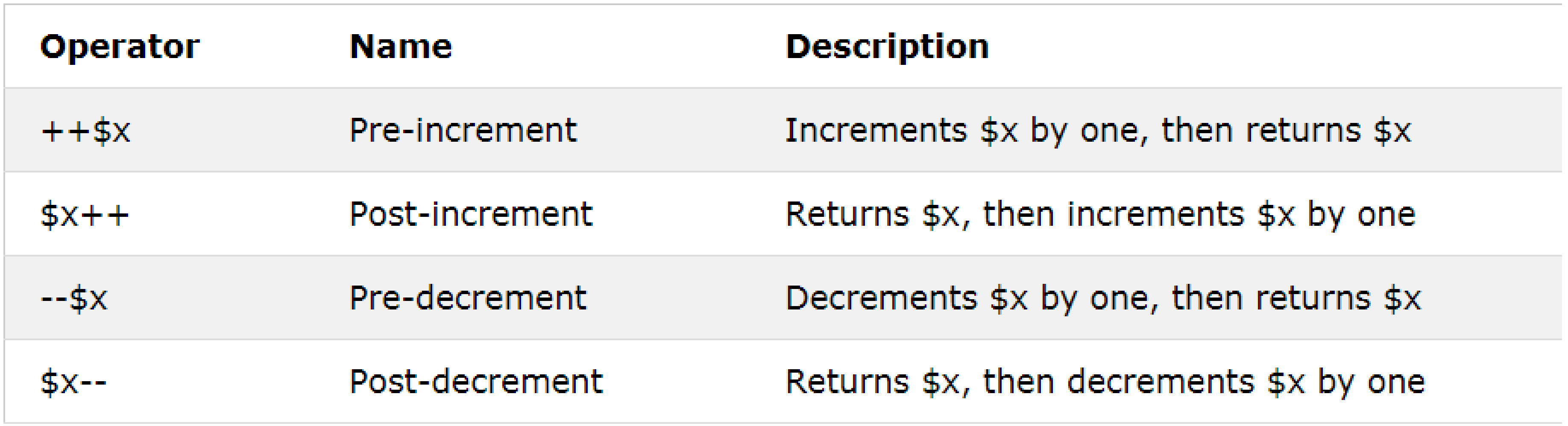


```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 100;
$y = "100";
var_dump($x === $y); // returns
false because types are not equal
echo "<br>";
var_dump($x == $y);
?>
</body> </html>
```

```
bool(false)
bool(true)
```

PHP Increment / Decrement Operators

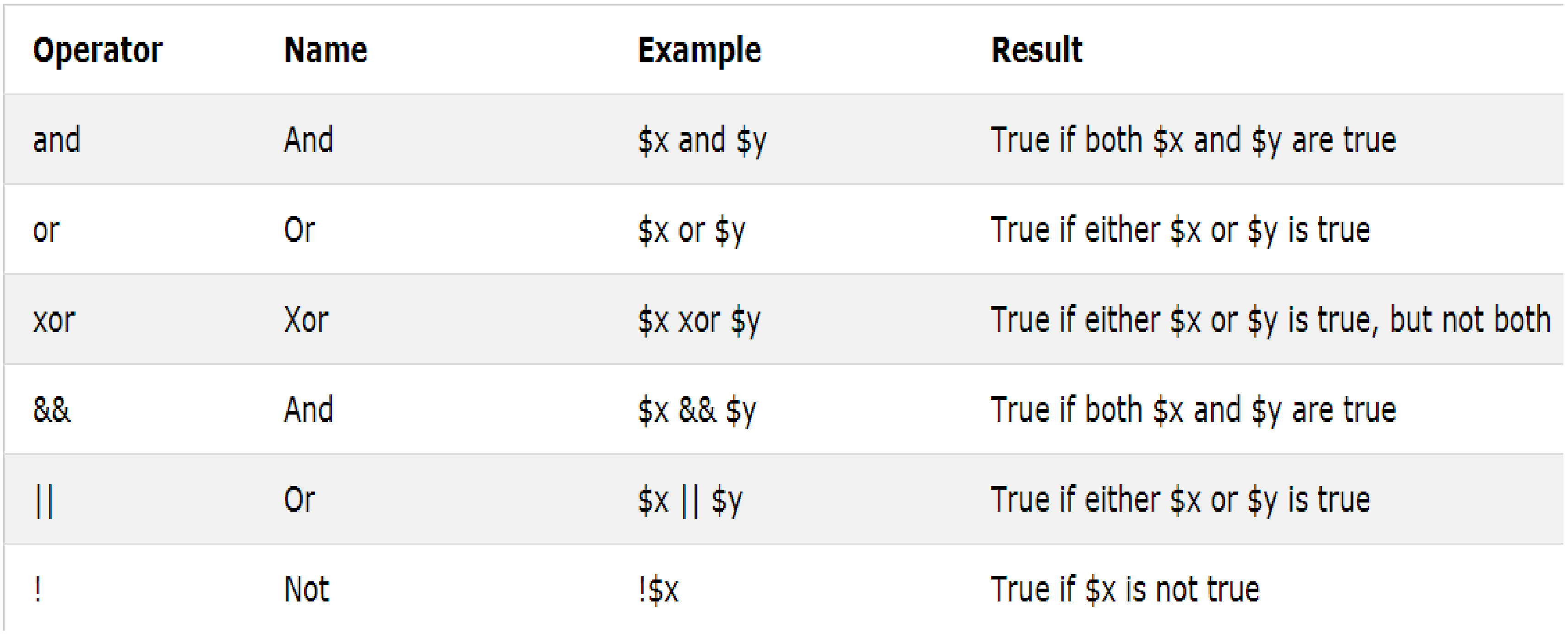
- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.



Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

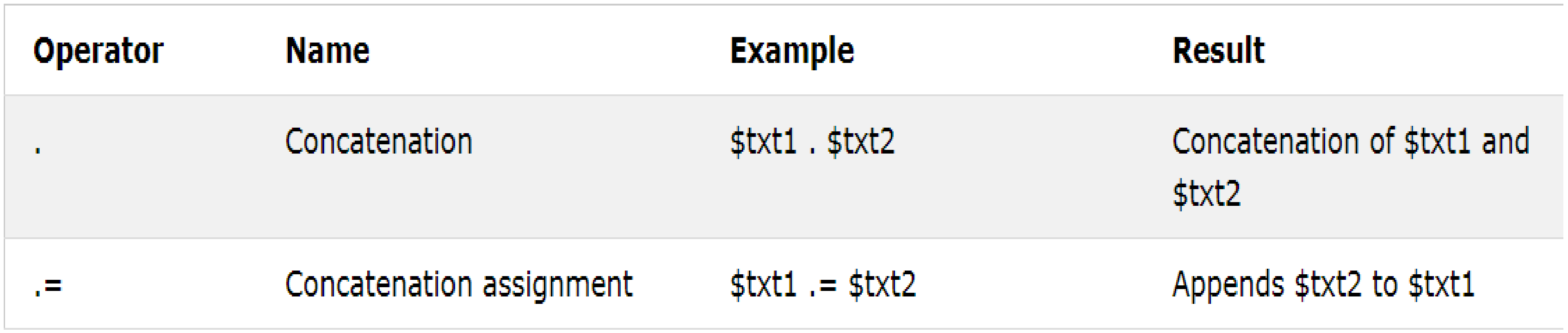
- The PHP logical operators are used to combine conditional statements.



Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

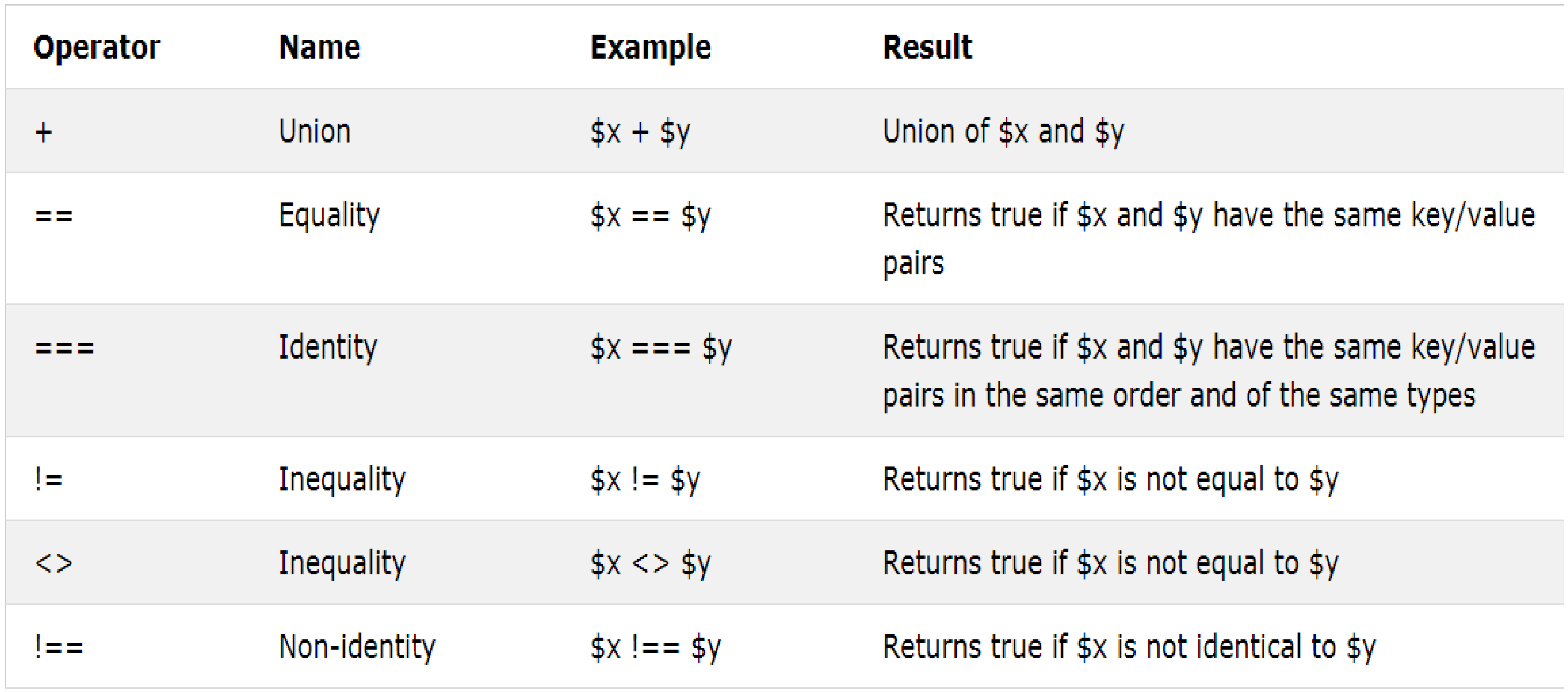
PHP String Operators

- PHP has two operators that are specially designed for strings.



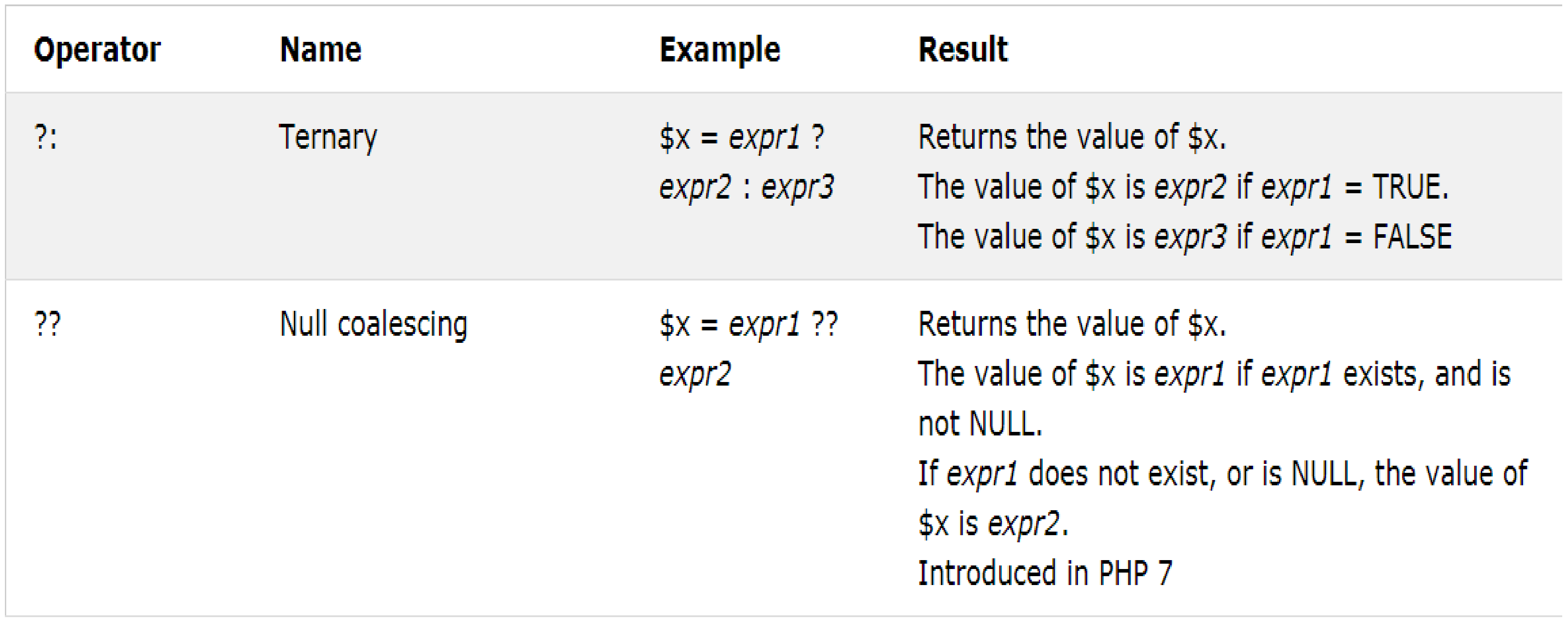
Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators



Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators



Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

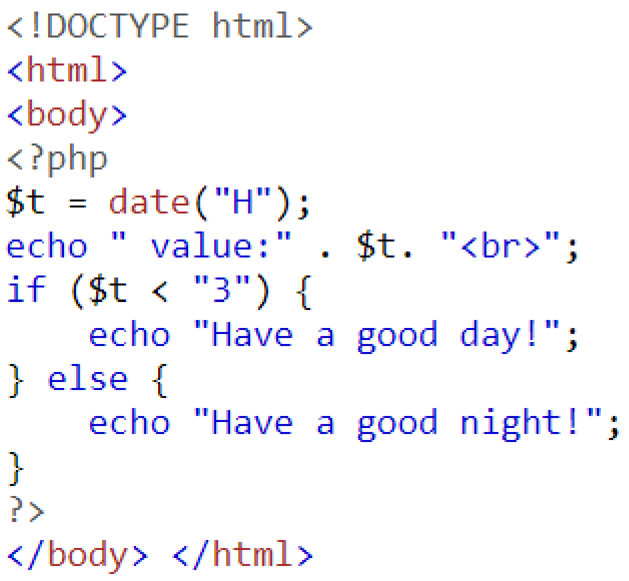
IF THEN ELSE

The if...else Statement

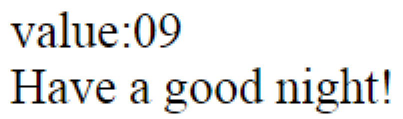
- The *if...else* statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

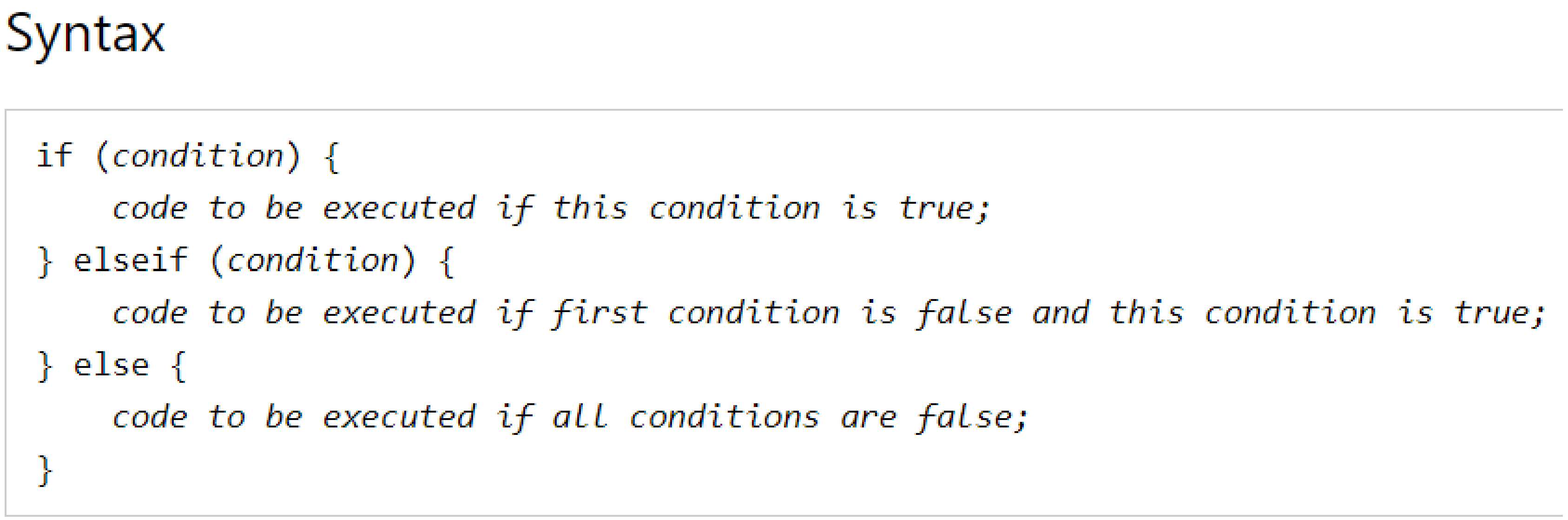


```
<!DOCTYPE html>
<html>
<body>
<?php
$t = date("H");
echo " value:" . $t. "<br>";
if ($t < "3") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
</body> </html>
```



```
value:09
Have a good night!
```

The if...elseif...else Statement



Syntax

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if first condition is false and this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

switch Statement

- Use the switch statement to **select one of many blocks of code to be executed**.

Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

LOOPS

End