PHP Programming Language

What is it?

- PHP is a scripting language designed for web development. It is used to produce dynamic web pages.
- Currently, PHP is one of the most widely used programming languages. Much of its syntax is borrowed from C, Java, and Perl with a couple of unique PHP-specific features.
- PHP can be embedded into HTML code and it generally runs on a web server.

Installation

- PHP can be installed via installation packages (direct from their website) or via development bundles WAMP, MAMP or XAMPP.
- Enabling php to run anywhere, you need to add the path of the php.exe binary to PATH environmental variable
- To test if php is running, simply run “php -v” you should be presented with its version, else make sure your PATH variable points to php.exe

PHP CLI

- PHP CLI is a command line interpreter for the PHP language. It is useful for testing PHP scripts from the shell. In this session we are using the PHP command line interpreter.
- We focus on the core of the PHP language. First Php program

simple.php

```
<?php

echo "this is PHP language\n";
```

To run it simply type:

> php simple.php
this is PHP language

PHP interactive shell

- Like Python or Ruby, PHP also has an interactive shell. It is useful to test small language constructs.

- > php -a
- Interactive mode enabled

```
php> print PHP_OS;
Linux
php> print PHP_VERSION;
7.2.11
```

- The PHP shell is invoked with the -a option of the php command. The shell uses the php> prompt.

PHP lexical structure

- Computer languages, like human languages, have a lexical structure. A source code of a PHP script consists of tokens.
- Tokens are atomic code elements.
- In PHP language, we have comments, variables, literals, operators, delimiters, and keywords.

PHP comments

- Comments are used by humans to clarify the source code. All comments in PHP follow the # character.

```
<?php
# comments.php
# Author Juan Tamad
# mysite 2019

echo "This is comments.php
script\n";
```

- Everything that follows the # character is ignored by the PHP interpreter.

```
// comments.php
// Author Juan Tamad
// mysite 2019

/*
comments.php
Author Juan Tamad
mysite 2019
*/
```

PHP white space

- White space in PHP is used to separate tokens in PHP source file.
- It is used to improve the readability of the source code.

```
$a=1;
$b = 2;
$c  =  3;
```

- The amount of space put between tokens is irrelevant for the PHP interpreter.
- It is based on the preferences and the style of a programmer.

```
$a = 1;
$b = 2; $c = 3;
$d
 =
 4;
```

PHP semicolon

- A semicolon is used to mark the end of a statement in PHP. It is mandatory.

```
$a = 34;
$b = $a * 34 - 34;
echo $a;
```

- Here we have three different PHP statements. The first is an assignment. It puts a value into the $a variable.
- The second one is an expression. The expression is evaluated and the output is given to the $b variable.
- The third one is a command. It prints the $a variable.

PHP variables

- A variable is an identifier, which holds a value. In programming we say that we assign a value to a variable. Technically speaking, a variable is a reference to a computer memory, where the value is stored.
- In PHP language, a variable can hold a string, a number, or various objects like a function or a class. Variables can be assigned different values over time.
- Variables in PHP consist of the $ character, called a sigil, and a label. A label can be created from alphanumeric characters and an underscore _ character.
- A variable cannot begin with a number. The PHP interpreter can then distinguish between a number and a variable more easily.

PHP variables examples

- These were valid PHP identifiers.
  $Value
  $value2
  $company_name

- These were examples of invalid PHP identifiers.
  $12Val
  exx
  $first-name

The variables are case sensitive. This means that $Price, $price, and $PRICE are three different identifiers

case.php

```
<?php

$number = 10;
$Number = 11;
$NUMBER = 12;

echo $number, $Number, $NUMBER;

echo "\n";
```

PHP constants

- A constant is an identifier for a value which cannot change during the execution of the script. By convention, constant identifiers are always uppercase.

- The following is a list of PHP compile time constants.

__CLASS__ __DIR__ __FILE__ __FUNCTION__ __METHOD__ __NAMESPACE__

constants.php

```php
<?php

define("SIZE", 300);
define("EDGE", 100);

#SIZE = 100;

echo SIZE;
echo EDGE;

echo "\n";
```

PHP literal

- A literal is any notation for representing a value within the PHP source code. Technically, a literal is assigned a value at compile time, while a variable is assigned at runtime.

```
$age = 29;
$nationality = "Hungarian";
```

- If we do not assign a literal to a variable, there is no way how we can work with it—it is dropped.

> php literals.php

```
Patrick 34
Luke 22
Jane 17
Rose 16
```

This is the output of the literals.php script.

literals.php

```
<?php

$name1 = "Jane ";
$age1 = 17;

$name2 = "Rose ";
$age2 = 16;

echo "Patrick 34\n";
echo "Luke 22\n";

echo $name1, $age1, "\n";
echo $name2, $age2, "\n";
```

PHP Operators

- An operator is a symbol used to perform an action on some value.

+ - * / % ** ++ -- ?: ?? = += -=

*= /= .= %= == != === !== <> > < >=

<= <=> && || ! xor or & ^ | ~ .

<< >>

PHP delimiters

- A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data stream.

```
$a = "PHP";
$b = 'Java';
```

- The single and double characters are used to mark the beginning and the end of a string.

```
function setDate($date) {
   $this->date = $data;
}
if ($a > $b) {
   echo "\$a is bigger than \$b";
}
```

- Parentheses are used to mark the function signature. The signature is the function parameters.

- Curly brackets are used to mark the beginning and the end of the function body. They are also used in flow control.

```
$a = array(1, 2, 3);
echo $a[1];
```

- The square brackets are used to mark the array index.

```
/*
 comments.php
 Author Juan Tamad
mysite 2019
*/
```

/* */ delimiters are used to provide C style comments in PHP.

```
<?php
// PHP code
```

The <?php delimiter is used to declare the start of PHP code.

PHP keywords

- A keyword is a reserved word in the PHP programming language. Keywords are used to perform a specific task in a computer program; for example, print a value, do repetitive tasks, or perform logical operations. A programmer cannot use a keyword as an ordinary variable.

The following is a list of PHP keywords.

abstract	and	array()	as	break
case	catch	class	clone	const
continue	declare	default	do	else
elseif	enddeclare	endfor	endforeach	endif
endswitch	endwhile	extends	final	for
foreach	function	global	goto	if
implements	interface	instanceof	namespace	new
or	private	protected	public	static
switch	throw	try	use	var
while	xor	yield	yield from	

Next we have other language constructs.

die()	echo()	empty()	exit()	eval()
include()	include_once()	isset()	list()	require()
require_once()	return()	print()	unset()	

PHP Basics

Running code

- All the PHP code is surrounded by two delimiters, <?php and ?>.

```
<?php

# PHP code
```

- PHP code is put between two delimiters.

PHP console output

- Output from our PHP scripts is sent to the console. Note that we say console because here we use the PHP_CLI command line interpreter. If we test these examples on the web, the output will be sent to the browser.

printing.php

```
<?php

$a = 23;
print $a;
```

echoing.php

```
<?php

$a = 23;
$b = 24;

echo $a, "\n", $b, "\n";
```

PHP command line arguments

- PHP scripts can receive command line arguments. They follow the name of the program. The $argv is an array holding all arguments of a PHP script. The $argc holds the number of arguments passed, including the name of the PHP script.

arguments.php

```
<?php

echo "There are $argc arguments\n";

for ($i=0; $i < $argc; $i++) {
    echo $argv[$i] . "\n";
}
```

```
$ php arguments.php 1 2 3
There are 4 arguments
arguments.php
1
2
3
```

PHP types

- PHP is a weakly typed language. It works with types, but the programmer does not specify them when declaring variables.
- A data type is a one of various types of data, as double, integer, or boolean. Values of a certain data type are from a specific range of values stating the possible values for that type, the operations that can be done on that type, and the way the values of that type are stored.

dynamic.php

```
<?php

$a = "Jane";
echo "$a \n";

$a = 12;
echo "$a \n";

$a = 56.4;
echo "$a \n";

$a = true;
echo "$a \n";
```

PHP types

- PHP works implicitly with data types. Programmers do not specify explicitly the data types.

  >php gettype.php
  double
  string
  integer
  array
  object

gettype.php

```php
<?php

$temperature = 12.4;
$name = "Jane";
$age = 17;
$values = array(1, 2, 3, 4, 5, 6);

class Being {};

$somebody = new Being();

echo gettype($temperature), "\n";
echo gettype($name), "\n";
echo gettype($age), "\n";
echo gettype($values), "\n";
echo gettype($somebody), "\n";
```

PHP constants

- In PHP, we can create constants. A constant is a name for a value that, unlike a variable, cannot be reassociated with a different value. We use the define() function to create constants in PHP.

constants.php

```
<?php

define("BLUE", "0000FF");

echo BLUE, "\n";

echo defined("BLUE");
echo "\n";
```

PHP constants

- Here we print some built-in PHP constants. For example, the PHP_OS constant prints the OS version on which the PHP was built.

predefined_constants.php

```
<?php

echo TRUE;
echo "\n";
echo PHP_VERSION;
echo "\n";
echo PHP_OS;
echo "\n";
echo __LINE__;
echo "\n";
echo __FILE__;
echo "\n";
echo DIRECTORY_SEPARATOR;
echo "\n";
echo PHP_DATADIR;
echo "\n";
```

PHP variable interpolation

- Variable interpolation is replacing variables with their values inside string literals. Another names for variable interpolation are: variable substitution or variable expansion.
- Output:

  > php interpolation.php

  Jane is 17 years old

  > php nointerpolation.php

  Jane is $age years old\n

interpolation.php

```
<?php

$age = 17;

echo "Jane is $age years old\n";
```

nointerpolation.php

```
<?php

$age = 17;

echo 'Jane is $age years old\n';
```

PHP including files

- PHP code is split in multiple files for bigger programs.
- We use the include statement to join various PHP files.

common.php

```php
<?php

define("VERSION", 1.12);

function get_max($x, $y) {
    if ($x > $y) {
        return $x;
    } else {
        return $y;
    }
}
```

myfile.php

```php
<?php

include "common.php";

echo "The version is " . VERSION . "\n";

$a = 5;
$b = 3;

echo get_max($a, $b), "\n";
```

PHP Data Types

Computer programs work with data. Tools to work with various data types are essential part of a modern computer language. A data type is a set of values and the allowable operations on those values.

PHP data types

List of PHP data types

PHP has eight data types:

Scalar types

- boolean
- integer
- float
- string

Compound types

- array
- object

Special types

- resources
- NULL

- Unlike in languages like Java, C, or Visual Basic, in PHP we do not
- A variable's type is determined at runtime by PHP.
- If we assign a string to a variable, it becomes a string variable

PHP Boolean values

- In PHP the boolean data type is a primitive data type having one of two values: True or False. This is a fundamental data type. Example

kid.php

```
<?php

$male = False;

$r = rand(0, 1);

$male = $r ? True: False;

if ($male) {
    echo "We will use name John\n";
} else {
    echo "We will use name Victoria\n";
}
```

boolean.php

```
<?php
class Object {};

var_dump((bool) "");
var_dump((bool) 0);
var_dump((bool) -1);
var_dump((bool) "PHP");
var_dump((bool) array(32));
var_dump((bool) array());
var_dump((bool) "false");
var_dump((bool) new Object());
var_dump((bool) NULL);
```

```
$ php boolean.php
bool(false)
bool(false)
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)
bool(true)
bool(false)
```

PHP integers

- Integers are a subset of the real numbers. They are written without a fraction or a decimal component. Integers fall within a set Z = {..., -2, -1, 0, 1, 2, ...}. Integers are infinite.
- In many computer languages, integers are primitive data types. Computers can practically work only with a subset of integer values, because computers have finite capacity.
- Integers are used to count discrete entities. We can have 3, 4, 6 humans, but we cannot have 3.33 humans. We can have 3.33 kilograms.

PHP integers

- Integers can be specified in four different notations in PHP: decimal, hexadecimal, octal, and binary.
- Octal values are preceded by 0, hexadecimal by 0x, and binary by 0b.

notation.php

```php
<?php

$decimal_var = 31;
$octal_var = 031;
$hexadecimal_var = 0x31;
$binary_var = 0b01001110;

echo "$decimal_var\n";
echo "$octal_var\n";
echo "$hexadecimal_var\n";
echo "$binary_var\n";
```

```
$ php notation.php
31
25
49
78
```

PHP floating point numbers

- Floating point numbers represent real numbers in computing.
- Real numbers measure continuous quantities, like weight, height, or speed.
- Floating point numbers in PHP can be larger than integers and they can have a decimal point.
- The size of a float is platform dependent.

floats.php

```php
<?php

$a = 1.245;
$b = 1.2e3;
$c = 2E-10;
$d = 1264275425335735;

var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
```

PHP floating point numbers

```
> php floats.php
float(1.245)
float(1200)
float(2.0E-10)
int(1264275425335735)

> php -a
Interactive mode enabled
php> echo 1/3;
0.33333333333333
php> $var = (0.33333333333333 == 1/3);
php> var_dump($var);
bool(false)
```

floats.php

```php
<?php

$a = 1.245;
$b = 1.2e3;
$c = 2E-10;
$d = 1264275425335735;

var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
```

PHP floating point numbers

- In this example, it is necessary to use floating point values.
- To get the speed, we divide the distance by the time.
- This is the output of the sprinter script. 36.474164133739 is a floating point number.
- It is often necessary to round floating point numbers.

rounding.php

```
<?php

$a = 1.4567;

echo round($a, 2) . "\n";
echo round($a, 3) . "\n";

echo sprintf("%0.3f", $a) . "\n" ;
```

PHP strings

- A string is a data type representing textual data in computer programs.
- We can use single quotes and double quotes to create string literals.
- The script outputs two strings to the console. The \n is a special sequence, a new line. The effect of this character is like if you hit the enter key when typing text.

strings.php

```
<?php

$a = "PHP ";
$b = 'Perl';

echo $a, $b;
echo "\n";
```

PHP arrays

- Array is a complex data type which handles a collection of elements. Each of the elements can be accessed by an index.
- In PHP, arrays are more complex. Arrays can be treated as arrays, lists, or dictionaries.
- In other words, arrays are all what in other languages we call arrays, lists, dictionaries.

arrays.php

```php
<?php

$names = [ "Jane", "Lucy", "Timea", "Beky", "Lenka" ];

print_r($names);
```

PHP arrays

- An array is created with the shorthand notation, where we use the square brackets.
- The elements of an array are separated with a comma character. The elements are strings.
- The print_r() function prints a human readable information about a variable to the console.

arrays.php

```
<?php

$names = [ "Jane", "Lucy", "Timea", "Beky", "Lenka" ];

print_r($names);
```

PHP arrays

```
> php arrays.php
Array
(
    [0] => Jane
    [1] => Lucy
    [2] => Timea
    [3] => Beky
    [4] => Lenka
)
```

- This is the output of the script. The numbers are indices by which we can access the array elements.

arrays.php

```php
<?php

$names = [ "Jane", "Lucy", "Timea", "Beky", "Lenka" ];

print_r($names);
```

PHP objects

Objects are user defined data types. Programmers can create their data types that fit their domain.

PHP resources

Resources are special data types. They hold a reference to an external resource. They are created by special functions. Resources are handlers to opened files, database connections, or image canvas areas.

PHP NULL

There is another special data type - NULL. Basically, the data type means non existent, not known or empty.

In PHP, a variable is NULL in three cases:

- it was not assigned a value
- it was assigned a special NULL constant
- it was unset with the unset() function

In our example, we have four variables. Three of them are considered to be NULL. We use the is_null() function to determine if the variable is NULL.

```
> php nulltype.php
$a is null
$b is null
$c is null
```

nulltype.php

```php
<?php

$a;
$b = NULL;

$c = 1;
unset($c);

$d = 2;

if (is_null($a)) echo "\$a is null\n";
if (is_null($b)) echo "\$b is null\n";
if (is_null($c)) echo "\$c is null\n";
if (is_null($d)) echo "\$d is null\n";
```

PHP type casting

- We often work with multiple data types at once. Converting one data type to another one is a common job in programming.
- Type conversion or typecasting refers to changing an entity of one data type into another.
- There are two types of conversion: implicit and explicit. Implicit type conversion, also known as coercion, is an automatic type conversion by the compiler. Example:

```
php> echo "45" + 12;
57
php> echo 12 + 12.4;
24.4
```

```
php> $a = 12.43;
php> var_dump($a);
float(12.43)
php> $a = (integer) $a;
php> var_dump($a);
int(12)
php> $a = (string) $a;
php> var_dump($a);
string(2) "12"
php> $a = (boolean) $a;
php> var_dump($a);
bool(true)
```

PHP Strings

Strings are very important data types in computer languages

PHP string literal

- A string literal is the notation for representing a string value within the text of a computer program.
- In PHP, strings can be created with single quotes, double quotes or using the heredoc or the nowdoc syntax.

literals.php

```php
<?php

$a = "PHP";
$b = 'PERL';

echo $a, $b;
```

string with a heredoc syntax

- The heredoc preserves the line breaks and other whitespace (including indentation) in the text.
- The heredoc is created with <<< followed by a delimiting identifier, followed, starting on the next line, by the text to be quoted, and then closed by the same identifier on its own line.
- The closing identifier must not be indented. It can only contain alphanumeric characters and underscores, and must start with a non-digit character or underscore.

string with a heredoc syntax

heredoc.php

```
<?php

$str = <<<TEXT
"That is just as I intended." Vautrin said. "You know quite well what
you are about. Good, my little eaglet! You are born to command, you
are strong, you stand firm on your feet, you are game! I respect you."
TEXT;

echo $str, "\n";
```

$ php heredoc.php
"That is just as I intended." Vautrin said. "You know quite well what you are about. Good, my little eaglet! You are born to command, you are strong, you stand firm on your feet, you are game! I respect you."

PHP interpolation

- Variables are interpolated in strings enclosed by double quotes.
- The $quantity variable is replaced with its value in the string output.

  > php interpolation.php

  There are 5 roses in the vase

  This is the output of the interpolation.php script.
- Curly braces can be used when the variable name is next to another character.

interpolation.php

```
<?php

$quantity = 5;

echo "There are $quantity roses in the vase\n";
```

curlybraces.php

```
<?php

$quantity = 5;
$item_name = "rose";

echo "There are $quantity {$item_name}s in the vase\n";
```

PHP string concatenation

PHP uses the dot . operator to concatenate strings.

```
php > echo "PHP " . "language\n";
PHP language
```

The example concatenates two strings.

```
php > $a = "Java ";
php > $a .= "language\n";
php > echo $a;
Java language
```

PHP also supports the .= compound operator.

PHP escape characters

An escape character is a single character designated to invoke an alternative interpretation on immediately subsequent characters in a character sequence.

```
php> echo "   bbb\raaa";
aaabbb
```

The carriage return \r is a control character for end of line return to the beginning of line.

strophe.php

```
<?php
echo "Incompatible, it don't matter though\n'cos someone's bound to hear my cry\n";
echo "Speak out if you do\nYou're not easy to find\n";
```

PHP escape characters

backslash.php

```
<?php

$text = "
\"That is just as I intended.\" Vautrin said. \"You know quite well what
you are about. Good, my little eaglet! You are born to command, you
are strong, you stand firm on your feet, you are game! I respect you.\"
";

echo $text;
```

In this example, we have a multiline text, which includes direct speech. The double quotes are escaped with the backslash character.

PHP escape characters

```
php> $var = 233;
php> echo "$var";
233
php> echo "\$var is $var";
$var is 233
```

- The dollar sign $ has also a special meaning in PHP; it denotes a variable. If a variable is used inside a string, it is interpolated, i.e. the value of the variable is used. To echo a variable name, we escape the $ character \$.

PHP string operations

- PHP has a large number of useful useful built-in functions that can be used for working with strings.

```
echo strlen("Eagle"); # prints 5
echo strtoupper("Eagle"); # prints EAGLE
echo strtolower("Eagle"); # prints eagle
```

- Here we use three functions. The strlen() function returns a number of characters in the string.
- The strtoupper() converts characters to uppercase letters, and the strtolower() converts characters to lowercase letters.

PHP string operations

- In our example, we have a string sentence. We calculate the absolute number of characters, number of alphabetic characters, digits and spaces in the sentence. To do this, we use the following functions: strlen(), ctype_alpha(), ctype_digit(), and ctype_space().

  >php letters.php

  There are 19 characters.

  There are 14 alphabetic characters.

  There are 2 digits.

  There are 3 spaces.

letters.php

```
<?php

$sentence = "There are 22 apples";

$alphas = 0;
$digits = 0;
$spaces = 0;

$length = strlen($sentence);

for ($i = 0; $i < $length; $i++) {

    $c = $sentence[$i];
    if (ctype_alpha($c)) $alphas++;
    if (ctype_digit($c)) $digits++;
    if (ctype_space($c)) $spaces++;

}

echo "There are $length characters.\n";
echo "There are $alphas alphabetic characters.\n";
echo "There are $digits digits.\n";
echo "There are $spaces spaces.\n";
```

substr() function

```
echo substr("PHP language", 0, 3); # prints PHP
echo substr("PHP language", -8); # prints language
```

- The function returns a part of a string. The first parameter is the specified string. The second parameter is the start of the substring.
- The third parameter is optional. It is the length of the returned substring. The default is to the return until the end of the string.

str_repeat

- The str_repeat() function repeats a string a specified number of times.
- We use the str_repeat() function to create two lines of the # character.

```
$ php repeat.php
##################
Project Neurea
Priority high
Security maximum
##################
```

- This is the output of the repeat.php script.

repeat.php

```
<?php

echo str_repeat("#", 18);
echo "\nProject Neurea\n";
echo "Priority high\n";
echo "Security maximum\n";
echo str_repeat("#", 18);
echo "\n";
```

Shuffling string

- In the next example, we will randomly modify a string.
- The str_shuffle() randomly shuffles a string.
- The str_shuffle() randomly shuffles a string.

```
>php shuffling.php
ZtCeoed
eodtCZe
toZeeCd
oCdeteZ
edtCZoe
tdeCeoZ
oeZdteC
```

shuffling.php

```
<?php

$string = "ZetCode";

echo str_shuffle($string), "\n";
echo str_shuffle($string), "\n";
echo str_shuffle($string), "\n";
echo str_shuffle($string), "\n";
echo str_shuffle($string), "\n";
echo str_shuffle($string), "\n";
echo str_shuffle($string), "\n";
```

Shuffling string

- The explode() function is used to split a string into parts. It returns an array of split string parts.
- We have integers within a string separated by comma character. We count the number of integers.

  $vals = explode(",", $nums);

- Here we split the text with the explode() function. The function will cut a string into pieces whenever it finds the dot , character.

exploding.php

```
<?php

$nums = "1,2,3,4,5,6,7,8,9,10,11";

$vals = explode(",", $nums);
$len = count($vals);

echo "There are $len numbers in the string\n";
```

Concatenating Strings

- We concatenate strings with the dot operator.

  $ php teams1.php

  Ajax Amsterdam - Inter Milano 2:3

  Real Madridi - AC Milano 3:3

  Dortmund - Sparta Praha 2:1

- The output is not optimal.

teams1.php

```
<?php

echo "Ajax Amsterdam" . " - " . "Inter Milano " . "2:3\n";
echo "Real Madridi" . " - " . "AC Milano " . "3:3\n";
echo "Dortmund" . " - " . "Sparta Praha ". "2:1\n";
```

Concatenating Strings

Improved Version

- We improve the output format with the str_pad() function. It adds a specified string (in our case a space) to the left of the string, to the right or to both sides.

  > php teams2.php

  Ajax Amsterdam - Inter Milano 2:3

  Real Madrid - AC Milano 3:3

  Dortmund - Sparta Praha 2:1

- We manage to give a nicer formatted output.

teams2.php

```php
<?php

$teams = array(
      array("Ajax Amsterdam", "Inter Milano"),
      array("Real Madrid", "AC Milano"),
      array("Dortmund", "Sparta Praha")
);

$results = array("2:3", "3:3", "2:1");

$i = 0;

foreach ($teams as $team) {
    echo str_pad($team[0], 14);
    echo str_pad("-", 3, " ", STR_PAD_BOTH);
    echo str_pad($team[1], 14);
    echo str_pad($results[$i], 3, " ", STR_PAD_LEFT);
    echo "\n";
    $i++;
}
```

Array of chars

- A string in PHP is an array of chars.
- We iterate through the string with the for loop.
- The size of the string is determined with the strlen() function.
- The ord() function returns the ASCII value of a character. We use the array index notation to get a character.

array_of_chars.php

```
<?php

$site = "zetcode.com";

for ($i=0; $i < strlen($site); $i++) {
    $o = ord($site[$i]);
    echo "$site[$i] has ASCII code $o\n";
}
```

Array of chars

```
> php array_of_chars.php
z has ASCII code 122
e has ASCII code 101
t has ASCII code 116
c has ASCII code 99
o has ASCII code 111
d has ASCII code 100
e has ASCII code 101
. has ASCII code 46
c has ASCII code 99
o has ASCII code 111
m has ASCII code 109
```

array_of_chars.php

```php
<?php

$site = "zetcode.com";

for ($i=0; $i < strlen($site); $i++) {
    $o = ord($site[$i]);
    echo "$site[$i] has ASCII code $o\n";
}
```

PHP string formatting

- String formatting or string interpolation is dynamic putting of various values into a string.

fruits.php

```
<?php

printf("There are %d oranges and %d apples in the basket.\n", 12, 32);
```

- We use the %d formatting specifier. The specifier expects an integer value to be passed.

  $ php fruits.php
  There are 12 oranges and 32 apples in the basket.

PHP string formatting

- The formatting specifier for a float value is %f and for a string %s.

  > php height.php
  Height: 172.300000 cm

- We might not like the fact that the number in the previous example has 6 decimal places by default.

- We can control the number of the decimal places in the formatting specifier.

height.php

```
<?php

printf("Height: %f %s\n", 172.3, "cm");
```

PHP string formatting

Other Formatting options:

- The first format works with hexadecimal numbers.
- The x character formats the number in hexadecimal notation.
- The o character shows the number in octal format.
- The e character shows the number in scientific format.

  > php formatting.php
  12c
  454
  3.000000e+5

formatting.php

```
<?php

# hexadecimal
printf("%x\n", 300);

# octal
printf("%o\n", 300);

# scientific
printf("%e\n", 300000);
```

PHP string formatting

- The next example prints three columns of numbers.

```
> php columns.php
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000
11 121 1331
```

columns.php

```
<?php

foreach (range(1,11) as $num) {
    echo $num , " ", $num*$num, " ",
         $num*$num*$num, "\n";
}
```

PHP string formatting

- To correct this, we use the width specifier. The width specifier defines the minimal width of the object. If the object is smaller than the width, it is filled with spaces.

columns2.php

```
<?php

foreach (range(1,11) as $num) {
    printf("%2d %3d %4d\n", $num, $num*$num, $num*$num*$num);
}
```

PHP string formatting

- Now the output looks OK. Number 2 says that the first column will be 2 characters wide.

```
> php columns2.php
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
10 100 1000
11 121 1331
```

columns2.php

```php
<?php

foreach (range(1,11) as $num) {
    printf("%2d %3d %4d\n", $num, $num*$num, $num*$num*$num);
}
```