PHP Piscine

Day 01

Staff 42 pedago@42.fr

Summary:

This document is the day01's subject for the PHP Piscine.

Contents

Chapter 1

Foreword

Catching a Porcupine

Father used to say
that if I were sitting,
waiting for a porcupine,
the time is always best
when the Milky Way turns back-
this is the time
when a porcupine returns.

Father also said
I should feel the wind.
He used to say
I should be careful
always to test
the direction of the wind.
The porcupine is not a thing
which will return, he'd say,
coming with the wind.
Rather, it moves
slant-wise, across it,
so that it can better
sniff the air and tell
if danger lurks ahead.

Father used to say
I should breathe softly
when sitting, waiting
for a porcupine.
It is a thing, he said,
which hears everything.
I must not even
make a rustling.
I must sit deadstill.

Father taught me

about the stars.
He used to say
that I should,
if sitting by a burrow,
I should watch the stars,
the places where they fell,
I should, above all,
watch them keenly,
for the places where stars fall,
he often taught,
really are the places
where porcupines can be caught.

These are translations based on the "Bleek Collection" |Xam [bushman] oral records taken down by the German linguist W.H. Bleek, and his assistant, Lucy Lloyd, in the 1870s. The "informants" listed are the |Xam people who related their poems and tales to Bleek and Lloyd. By the end of the century, the |Xam had been effectively exterminated; nobody on earth today can speak their language.

Chapter 2

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get **-42**, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We **will not** take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called **Google / the Internet / http://www.php.net / ....**
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject …
- By Odin, by Thor ! Use your brain !!!

Chapter 3

Exercise 00 : HW

Turn-in directory : ex00/
Files to turn in: hw.php
Allowed functions: The whole standard PHP library

Reminder: PHP is really easy. It's like C, except that we do not declare the variables. You just place a dollar sign in front of them, they are not typed, and there is no main. The rest is just a detail.

Today, we will keep working in PHP with the command line. Start by creating a small program, quite simple, called **hw.php**. This program must greet the world with its famous message.

```
$> ./hw.php
Hello World
$>
```

Chapter 4

Exercise 01 : mlX

Turn-in directory : ex01/

Files to turn in: mlx.php

Allowed functions: The whole standard PHP library

Some of you might be super comfortable with the minilibX but we are sorry to inform you that there are no PHP bindings that you can use here. This exercise has nothing to do with graphics nor with maths. Nope, what you need is to create a program that can display **1000** times the letter **X**, a newline, and with the constraint that this program must be written with less than **100** characters.

```
$> ls -la mlx.php
-rwxr-xr-x  1 boulon  users  92 Mar 12 11:54 mlx.php
$> ./mlx.php
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

[TL;DP]

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
$>
```

Chapter 5

Exercise 02 : to the Divine

Turn-in directory : ex02/
Files to turn in: oddeven.php
Allowed functions: The whole standard PHP library

As the Wise Old Man used to say, it's thanks to Olympia, the detergent of the Gods that the laundry is so soft and smells so good. But if you think about it, there was only 1 chance out of 2 to wash the right pile of laundry with it. It depended on whether it had an even or an odd number. Create a program in PHP that will kindly ask the user for a pile number, and that will inform them if it's **even** (therefore washed with Olympia) or if it's **odd**.

```
$> ./oddeven.php
Enter a number: 42
The number 42 is even
Enter a number: 0
The number 0 is even
Enter a number:
'' is not a number
Enter a number: toto
'toto' is not a number
Enter a number: 21
The number 21 is odd
Enter a number: 99cosmos
'99cosmos' is not a number
Enter a number: ^D
$>
```

Pay attention to the examples: the exact messages, the special cases, etc. You must be able to exit your program with "CTRL-D". The readline library is not part of the PHP standard lib.

Chapter 6

Exercise 03 : ft_split

Turn-in directory : ex03/
Files to turn in: ft_split.php
Allowed functions: The whole standard PHP library

Create the **ft_split** function. It will take a string as an argument, and will return a sorted array with the different words, initially separated by one or more spaces in the original string. Your **ft_split.php** will be included in a php test file.

```
<?PHP
include("ft_split.php");

print_r(ft_split("Hello    World AAA"));
?>
```

```
$> ./main.php
Array
(
    [0] => AAA
    [1] => Hello
    [2] => World
)
$>
```

Chapter 7

Exercise 04 : aff_param

Turn-in directory : ex04/
Files to turn in: aff_param.php
Allowed functions: The whole standard PHP library

This very basic program displays its command line arguments in the order it received them. The name of the program must not be displayed.

```
$> ./aff_param.php
$> ./aff_param.php foo bar hello world quax
foo
bar
hello
world
quax
$>
```

Chapter 8

Exercise 05 : epur_str

Turn-in directory : ex05/
Files to turn in: epur_str.php
Allowed functions: The whole standard PHP library

This program takes one unique argument and removes all the spaces at the beginning and at the end of the string. It should also reduce the spaces between each word to one single space. There will be only spaces, no tabulations or anything.

```
$> ./epur_str.php
$> ./epur_str.php "Hello,     how do  you   do     ?" | cat -e
Hello, how do you do ?$
$> ./epur_str.php "   Hello World    " | cat -e
Hello World$
$>
```

Chapter 9

Exercise 06 : ssap

Turn-in directory : ex06/
Files to turn in: ssap.php
Allowed functions: The whole standard PHP library

Do not confuse it with the enterprise management software SAP.
This exercise will let you mix the previous two exercises. Your program must display all the words contained in all of the arguments, sorted.

```
$> ./ssap.php
$> ./ssap.php foo bar
bar
foo
$> ./ssap.php foo bar "yo man" "Here is my, two words" Xibul
Here
Xibul
bar
foo
is
man
my,
two
words
yo
$>
```

Chapter 10

Exercise 07 : rostring

Turn-in directory : ex07/
Files to turn in: rostring.php
Allowed functions: The whole standard PHP library

Your program will take a string as argument, and will place the first word (space separated portion of the string) at the last position. The string must then be displayed with only one space between each word.

```
$> ./rostring.php
$> ./rostring.php sdfkjsdkl sdkjfskljdf
sdfkjsdkl
$> ./rostring.php "hello world  aaa" fslkdjf
world aaa hello
$>
```

Chapter 11

Exercise 08 : ft_is_sort

Turn-in directory : ex08/
Files to turn in: ft_is_sort.php
Allowed functions: The whole standard PHP library

You need to create a little function that will return true or false according to whether the array passed as argument is sorted or not.

```php
<?PHP
include("ft_is_sort.php");

$tab = array("!/@#;^", "42", "Hello World", "hi", "zZzZzZz");
$tab[] = "What are we doing now ?";

if (ft_is_sort($tab))
  echo "The array is sorted\n";
else
  echo "The array is not sorted\n";
?>
```

```
$> ./main.php
The array is not sorted
$>
```

Chapter 12

Exercise 09 : ssap - the return

Turn-in directory : ex09/
Files to turn in: ssap2.php
Allowed functions: The whole standard PHP library

Get back to your **ssap.php**. You need to do the same thing again (take all the words from all the parameters and sort them) but you need to change the sorting rule: now it will have to be case insensitive and place all the characters in alphabetical order first, then numbers, and finally all the other characters, each of these groups following the ASCII order.

```
$> ./ssap2.php
$> ./ssap2.php toto tutu 4234 "_hop XXX" ## "1948372 AhAhAh"
AhAhAh
toto
tutu
XXX
1948372
4234
##
_hop
$>
```

Chapter 13

Exercise 10 : do_op

Turn-in directory : ex10/
Files to turn in: do_op.php
Allowed functions: The whole standard PHP library

This PHP program will take 3 arguments. The first and third ones are numbers. The second is an arithmetic operation amongst : **+**, **-**, *****, **/**, **%**. You need to make this operation and display the result. The program does not need to manage errors, except the number of arguments given.
There can be spaces and tabulations in all of the arguments.

```
$> ./do_op.php
Incorrect Parameters
$> ./do_op.php 1 + 3
4
$> ./do_op.php " 1" " +" " 3"
4
$> ./do_op.php " 1" " *" " 3"
3
$> ./do_op.php 42 "% " 3
0
$>
```

Note: respect the error message.

Chapter 14

Exercise 11 : do_op_2

Turn-in directory : ex11/
Files to turn in: do_op_2.php
Allowed functions: The whole standard PHP library

This time, your program will receive only one argument. It will contain the whole calculation that needs to be done. It will always be under the format of **number** - **operator** - **number**. A new error message "Syntax Error" will now complete the prior message in case the syntax isn't correct.
There can be either zero or multiple spaces between the numbers and the operator. The expected result is the same.

```
$> ./do_op_2.php
Incorrect Parameters
$> ./do_op_2.php toto
Syntax Error
$> ./do_op_2.php "42*2"
84
$> ./do_op_2.php "  42 / 2 "
21
$> ./do_op_2.php "six6*7sept"
Syntax Error
$> ./do_op_2.php '`rm -rf ~/`;'
Syntax Error
$>
```

Chapter 15

Exercise 12 : search_it!

Turn-in directory : ex12/
Files to turn in: search_it!.php
Allowed functions: The whole standard PHP library

Your program will take as first argument a string containing a key. It will then search amongst an unlimited number of '**key:value**' pairs the value corresponding to the given key and display it.

```
$> ./search_it!.php
$> ./search_it!.php toto
$> ./search_it!.php toto "key1:val1" "key2:val2" "toto:42"
42
$> ./search_it!.php toto "toto:val1" "key2:val2" "toto:42"
42
$> ./search_it!.php "toto" "key1:val1" "key2:val2" "0:hop"
$> ./search_it!.php "0" "key1:val1" "key2:val2" "0:hop"
hop
$>
```

Chapter 16

Exercise 13 : The hesitating agent

Turn-in directory : `ex13/`
Files to turn in: `agent_stats.php`
Allowed functions: `The whole standard PHP library`

After yesterday's agent (the one that shrank), here is the one that hesitates and needs to make choices. Among the resources of the day, you have a few files of peer-grades, take a look at them before starting your program.
You will have to implement the following options:

- The **average** option will calculate the average grade given by peers (not by Moulinette then)
- The **average_user** option will calculate the average grade given by peers per user ordered by alphabetical order.
- The **moulinette_variance** will calculate the difference between the average grade given by peers and the grade given by Moulinette.

The file will be read on the standard input, and the option will be passed as an argument to the program. If the user has no grade for the last two options, the line is not displayed.

See the examples in the next page.

```
$> cat peer_notes_1.csv | ./agent_stats.php
$> cat peer_notes_1.csv | ./agent_stats.php average
9.8621262458472
$> cat peer_notes_1.csv | ./agent_stats.php average_user
adam_e:9.0555555555556
bertrand_y:7.9473684210526
bruce_w:9.0434782608696
clark_k:10.464285714286
david_a:8.68
dexter_m:8.9

     [.....]

sandra_n:11.181818181818
steve_j:11.5
trevor_r:6.1052631578947
$>
```

```
$> cat peer_notes_1.csv | ./agent_stats.php moulinette_variance
adam_e:3.0555555555556
bertrand_y:-1.0526315789474
bruce_w:-9.9565217391304
clark_k:0.46428571428571

     [.....]

steve_j:10.5
trevor_r:-12.894736842105
$>
```