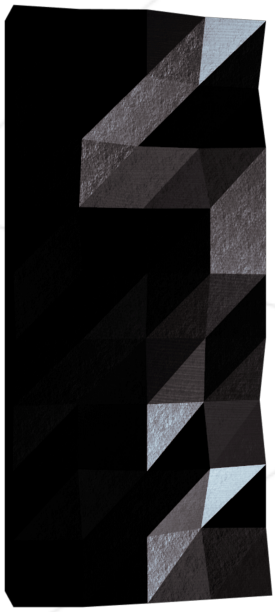

PHP Piscine

Day 01

Staff 42 pedago@42.fr

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

Contents

I	Foreword	2
II	General Instructions	4
III	Exercise 00 : HW	5
IV	Exercise 01 : mlX	6
V	Exercise 02 : to the Divine	7
VI	Exercise 03 : ft_split	8
VII	Exercise 04 : aff_param	9
VIII	Exercise 05 : epur_str	10
IX	Exercise 06 : ssap	11
X	Exercise 07 : rostring	12
XI	Exercise 08 : ft_is_sort	13
XII	Exercise 09 : ssap - the return -	14
XIII	Exercise 10 : do_op	15
XIV	Exercise 11 : do_op_2	16
XV	Exercise 12 : search_it!	17
XVI	Exercise 13 : The hesitating agent	18

Chapter I

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 II

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 III

Exercise 00 : HW

	Exercise 00
HW	
Turn-in directory : $ex00/$	
Files to turn in : `hw.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

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 in no main. The rest, is -almost- a detail.

Today, we will stay on PHP in 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
$>
```

Note: MMORPG Day (Massive Moulinette Online Rules PHP Geniuses)

Chapter IV

Exercise 01 : mlX

	Exercise 01
mlX	
Turn-in directory : $ex01/$	
Files to turn in : `mlx.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Since you are all super comfortable with **minilibX**, I am sorry to inform you that there are no **PHP** binding that you can use here. This exercise has nothing to do with **graphics** nor with **maths**. Nope, what you need to create, is a program that can display 1000 times the letter X, a newline, and with the constraint that it cannot go over 100 chars.

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

[TL;DP]

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
$>
```

Chapter V

Exercise 02 : to the Divine

	Exercise 02
to the Divine	
Turn-in directory : $ex02/$	
Files to turn in : `oddeven.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

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 of you a pile number, and that will inform you 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 example, in particular spaces, the uppercases and the exact messages. At the end, it a 'CTRL-D' to exit. And the readline library is not a part of the standard PHP library.

Chapter VI

Exercise 03 : ft_split

	Exercise 03
ft_split	
Turn-in directory : $ex03/$	
Files to turn in : `ft_split.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

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

```php
<?PHP

include("ft_split.php");

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

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

Chapter VII

Exercise 04 : aff_param

	Exercise 04
aff_param	
Turn-in directory : $ex04/$	
Files to turn in : `aff_param.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Very basic, this program displays its command line arguments in the order received. The name of the program isn't displayed.

```
$> ./aff_param.php
$> ./aff_param.php toto ahah foo bar quax
toto
ahah
foo
bar
quax
$>
```

Chapter VIII

Exercise 05 : epur_str

	Exercise 05
epur_str	
Turn-in directory : $ex05/$	
Files to turn in : `epur_str.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

This program takes one unique argument and reduces to a single space between each word, and none at the beginning and at the end of the string. There are only spaces, no tabulation or anything.

```
$> ./epur_str.php
$> ./epur_str.php "Hello,     how do  you   do     ?"
Hello, how do you do ?
$> ./epur_str.php "   Hello World    "
Hello World
$>
```

Chapter IX

Exercise 06 : ssap

	Exercise 06
ssap	
Turn-in directory : $ex06/$	
Files to turn in : `ssap.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Do not confuse it with the enterprise management software SAP, it is for you the chance to mix the prior two exercises. The sum of words contained in all the arguments (except the name of the program itself) are splited, sorted and displayed.

```
$> ./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 X

Exercise 07 : rostring

	Exercise 07
rostring	
Turn-in directory : $ex07/$	
Files to turn in : `rostring.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Your program will take a string as argument, and will place the first word (space separated) at the last spot. The whole thing is then re- presented, with 1 space only between each word.

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

Chapter XI

Exercise 08 : ft_is_sort

	Exercise 08
ft_is_sort	
Turn-in directory : *ex*08/	
Files to turn in : `ft_is_sort.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

You need to create a little function that will reply 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 XII

Exercise 09 : ssap - the return -

	Exercise 09
ssap - the return -	
Turn-in directory : *ex*09/	
Files to turn in : `ssap2.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

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 need to be case insensitive and place all the characters in alphabetical order first, then numbers, and finally all the other characters, each in the following 3 groups following the ASCII order.

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

Chapter XIII

Exercise 10 : do_op

	Exercise 10
do_op	
Turn-in directory : $ex10/$	
Files to turn in : `do_op.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

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

Exercise 11 : do_op_2

	Exercise 11
do_op_2	
Turn-in directory : $ex11/$	
Files to turn in : `do_op_2.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

This time, only 1 argument is on the menu. That one contains the whole calculation that needs to be done. This calculation will always be under the following format *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 no space between the numbers and the operator, or there can be many. The expected results 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 XV

Exercise 12 : search_it!

	Exercise 12
search_it!	
Turn-in directory : $ex12/$	
Files to turn in : `search_it!.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

Your goal is to create a program that will display the corresponding value to a key given as first argument amongst an unlimited number of couples formated like this: “key:value” given as following arguments.

```
$> ./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 XVI

Exercise 13 : The hesitating agent

	Exercise 13
The hesitating agent	
Turn-in directory : $ex13/$	
Files to turn in : `agent_stats.php`	
Allowed functions : `The whole standard PHP library`	
Notes : `n/a`	

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-notes. The idea is for you to choose precisely how to use them. Here are a few solutions:

- the "average" option will calculate the average grade without Moulinette.
- the "average_user" option will calculate the average grade per user ordered by alphabetical order.
- the "moulinette_variance" will calculate the average grade per user of the difference between a grade received by your peer and by Moulinette.

The grade file will be read on the standard input, and the option is passed as an argument to the program. In cases 2 and 3, if a user has no grade, it's not displayed.

```
$> cat peer_notes_1.csv | ./agent_stats.php
$> cat peer_notes_1.csv | ./agent_stats.php moyenne
9.8621262458472
$> cat peer_notes_1.csv | ./agent_stats.php moyenne_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 ecart_moulinette
adam_e:3.0555555555556
bertrand_y:-1.0526315789474
bruce_w:-9.9565217391304
clark_k:0.46428571428571

    [.....]

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