



# CodeInActionLab

## Python-04

*Summary: This document is the subject for the Python-04 module of the CodeInActionLab @ aba2020.*

*Version: 1.0*

# Contents

<b>I</b>	<b>Instructions</b>	<b>2</b>
<b>II</b>	<b>Foreword</b>	<b>3</b>
<b>III</b>	<b>Exercise 00 : filemanager</b>	<b>4</b>
<b>IV</b>	<b>Exercise 01 : filemanager</b>	<b>5</b>
<b>V</b>	<b>Exercise 02 : filemanager</b>	<b>6</b>
<b>VI</b>	<b>Exercise 03 : filemanager</b>	<b>7</b>
<b>VII</b>	<b>Exercise 04 : rectangle</b>	<b>8</b>
<b>VIII</b>	<b>Submission and peer-evaluation</b>	<b>9</b>

# Chapter I

## Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called 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.
- Moulinette is not very open-minded.
- 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.
- Using a forbidden function is considered cheating. Cheaters get **-42**, and this grade is non-negotiable.
- If your program doesn't compile, you'll get 0.
- You cannot leave any additional file in your directory 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 / man / the Internet / ....**
- 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 II

## Foreword

Here is a discuss extract from the Silicon Valley serie:

- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emacs.
- Oh, God, help us! Okay, uh you know what? I just don't think this is going to work. I'm so sorry. Uh, I mean like, what, we're going to bring kids into this world with that over their heads? That's not really fair to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no way I'm going to be with someone who uses spaces over tabs.
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! - (DOOR SLAMS) - (BANGING)

. . .


(RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time. I'm okay, though.
- See you around, Richard.
- Just making a point.

Hopefully, you are not forced to use emacs and your space bar to complete the following exercises.

# Chapter III

## Exercise 00 : filemanager

	Exercise 00
Turn-in directory : <i>ex00/</i>	
Files to turn in : <code>filemanager.py</code>	
Allowed functions : <code>print</code>	


- Write a program that reads "words.txt" and prints only the words with **more than 20 characters** (not counting whitespaces).

Here's how it should be prototyped:

```
class Dictionary:
def printWords():
```

# Chapter IV

## Exercise 01 : filemanager

	Exercise 01
filemanager	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <b>filemanager.py</b>	
Allowed functions : <b>None</b>	


- Write a program that does the same as the first exercise, but printing on the file **long<sub>w</sub>ords.txt**.

Here's how it should be prototyped:

```
class Dictionary:
def writeWords():
```

# Chapter V

## Exercise 02 : filemanager

	Exercise 02
filemanager	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>filemanager.py</b>	
Allowed functions : <b>None</b>	


- Write a program that does the same as the second exercise, but letting the user insert the filename and catching exceptions (in case the file cannot be opened).

Here's how it should be prototyped:

```
class Dictionary:
def writeWords():
```

# Chapter VI

## Exercise 03 : filemanager


	Exercise 03
filemanager	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <b>filemanager.py</b>	
Allowed functions : None	

- Write a program that reads **words.txt**, constructs a dictionary whose keys are integers and such that the value associated to **N** is the number of words in **words.txt** whose length is **N**.



# Chapter VII

## Exercise 04 : rectangle

	Exercise 04
rectangle	
Turn-in directory : <i>ex04/</i>	
Files to turn in : <b>rectangle.py</b>	
Allowed functions : <b>None</b>	

- Write a program that extends exercise 4 by dumping the dictionary using pickle.

# Chapter VIII

## Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.



You need to return only the files requested by the subject of this project.