CodeInActionLab

Python-04

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

Version: 1.2

Contents

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 the easiest to the 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` series:

```
- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emac.
- 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 : print_words

	Exercise 00
Turn-in directory : *ex*00/	
Files to turn in : `print_words.py`	
Allowed functions : `exit, len, open, print`	

- Write a program that reads "words.txt" and prints only the words with **more than 20 characters** (not counting whitespaces).
  - If the file does not exist, print `Error! File not found` and exit.

Here's how it should be prototyped:

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

Chapter IV

Exercise 01 : write_file

	Exercise 01
write_file	
Turn-in directory : *ex*01/	
Files to turn in : `write_file.py`	
Allowed functions : `exit, len, open, print`	

- Write a program that does the same as the first exercise, but print on the file **long_words.txt**.
  - If the file does not exist, print `Error! File not found` and exit.

Here's how it should be prototyped:

```
def writeWords():
```

```
You have to copy the same class of exercise 00 and add the new
method.
```

Chapter V

Exercise 02 : filemanager

	Exercise 02
filemanager	
Turn-in directory : *ex*02/	
Files to turn in : `filemanager.py`	
Allowed functions : `exit, input, len, open, print`	

- 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).
  - If the file does not exist, print `Error! File not found` and exit.
  - If the file does not end with .txt, print `Error! File not found` and exit.

Here's how it should be prototyped:

```
def customWriteWords():
```

Chapter VI

Exercise 03 : dictionary

	Exercise 03
dictionary	
Turn-in directory : *ex*03/	
Files to turn in : `dictionary.py`	
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 : filemanager

	Exercise 04
filemanager	
Turn-in directory : *ex*04/	
Files to turn in : `filemanager.py`	
Allowed functions : `None`	

- 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.`