Python Bootcamp {2019}

Kirirom Institute of Technology

EXERCISES WEEK 3

EXERCISE 41

Description :

You will write a function that print the current path of your program folder, then you will return it as a string.

Requirements :

- Program name : **41_current_path.py**
- Function name : **current_path**
- Directory : **week03/ex** folder

Hint :

- os library
- os path

EXERCISE 42

Description :

You will write a function that return a tuple list with all files and folders inside the current path. For each tuple, the first value will be the file or folder name. The second value will be the kind of file ('Folder' or 'File').

Python Bootcamp {2019}

EXAMPLE :

If the current folder path contain 3 FOLDERS (folder_01, folder_02 and folder_3) and 2 FILES (file_01, file_02)

The program will return :

[('folder_01','Folder'),('folder_02','Folder'),('folder_03','Folder'),('file_01','File'),('file_02','File')]

You must first return all the folders, sorted alphabetically and then the files sorted alphabetically as well.

Your program will not return the name of this program.
If not folders / files found, you will return an empty list.

The program will return :

[]

Requirements :

- Program name : **42_current_folder.py**
- Function name : **current_folder**
- Directory : **week03/ex** folder

Hint :

- ❖ os library
- ❖ os.path

Python Bootcamp {2019}

EXERCISE 43

Description :

You will write a function that take a filename as argument (string). The filename represent the name of the file your program will read. First you will make sure that the file exist, then you will print the content and finally you will return the value as string.

If the file is not valid (folder) or doesn't exist :

You will print 'Invalid FILENAME'
Then, you will return an empty string.

Requirements :

- Program name : **43_read_file.py**
- Function name : **read_file**
- Directory : **week03/ex** folder

Hint :

- ❖ Make sure your program doesn't crash if the filename is not valid.
- ❖ Make sure that after open a file, you close it.

EXERCISE 44

Description :

You will write a function that take two arguments: The first argument is a filename, the second is the content. Your program will create a new file inside the current directory with the name you specify. Then it will write the content you specify in the second argument. Finally your program will close the file and return 1

If you try to replace a file that already exist, you will ask confirmation of the user :

"Are you sure you want to replace <FILENAME>? [Y/N]"

If the user write Y, you will create the new file and return 1
If the user write N, you will just return 0
If the user write anything else, you will print :

"Invalid Option" and then print the confirmation message again:
"Are you sure you want to replace <FILENAME>? [Y/N]"

Be careful with this program and don't delete your work!

Requirements :

- Program name : **44_write_file.py**
- Function name : **write_file**
- Directory : **week03/ex** folder

EXERCISE 45

Description :

You will create a program that take a list of string as argument, that represents folders names. Then, for each, you will create a folder with the corresponding name. Before create anything, you will check that the folders doesn't already exists. If they, you will ask:

"Are you sure you want to replace <FOLDER_NAME>? [Y/N]"

If the user enter anything that is not Y or N, you will write:

"Invalid Option" and then print the confirmation message again:
"Are you sure you want to replace <FOLDER_NAME>? [Y/N]"

Make sure that you ask for EVERY folders that already exist only.
Be careful with this program and don't delete your work!

At the end, if your program did create any new folder you will return 1 Else you will return 0. If the list of folder name is empty, your program will also return 0.

Requirements :

- Program name : **45_auto_folder.py**
- Function name : **auto_folder**
- Directory : **week03/ex** folder

Hint :

- ❖ os library

EXERCISE 46

Description :

You will create a function that has the power to delete any file or any folder that is specify. Your function will take one parameter (filename) that can be a Folder or a File. Then, the function will ask for confirmation:

"Are you sure you want to delete <FILENAME>? [Y/N]"

If the user write anything that is not Y or N you will print:

"Invalid Option" and then print the confirmation message again:
"Are you sure you want to delete <FILENAME>? [Y/N]"

If a file has been deleted, your function will return 1, else return 0.

Be careful with this program and don't delete your work!

Requirements :

- Program name : **46_delete_file.py**
- Function name : **delete_file**
- Directory : **week03/ex** folder

Python Bootcamp {2019}

Kirirom Institute of Technology

EXERCISE 47

Description :

You will create a program that will allow to keep log of your messages inside a file. To do so, you will use the append function (it's allow you to add content at the end of an existing file, if the file doesn't exist, it will be created automatically).

Your function will take one argument in parameter (filename) that will be the given name of your log file.

Then you will have a loop that will stop ONLY if the user enter 'EXIT' The loop will contain an input that will display: "$: "

Anything (except EXIT) that will be enter by the user, will create a new line inside the file. Also, before the line, the current date and time will be display in this format : [DD-MM-YYYY hh:mm:ss]

EXAMPLE :

"$: The Week 3 just started"
"$: It's time to eat! I'm hungry"
"$: The bootcamp is finish for today! "

EXPECTED OUTPUT IN THE FILE :

[03/06/2019 11:00:00] The Week 3 just started
[03/06/2019 12:00:05] It's time to eat! I'm hungry
[03/06/2019 17:00:00] The bootcamp is finish for today!

The date and time must correspond to the time you wrote the message.

Requirements :

- Program name : **47_append_file.py**
- Function name : **append_file**
- Directory: **week03/ex** folder

Python Bootcamp {2019}

Kirirom Institute of Technology

EXERCISE 48

Description :

You will create a function that convert a JSON Array into a TSV File. The JSON file will look like this one:

```
[{
    "title": "week 01",
    "content": "python syntax",
    "difficulty": "very easy"
  },
  {
    "title": "week 02",
    "content": "python data manipulation",
    "difficulty": "easy"
  },
  {
    "title": "week 03",
    "content": "python files and requests",
    "difficulty": "intermediate"
  },
  {
    "title": "week 04",
    "content": "python class and advanced concepts",
    "difficulty": "hard"
  }
]
```

Once you will convert it, if you open the new created file with excel or google sheets, it will look like this one:

	A	B	C
1	title	content	difficulty
2	week01	python syntax	very easy
3	week02	python data manipulation	easy
4	week03	python files and requests	intermediate
5	week04	python class and advanced concepts	hard

Python Bootcamp {2019}

The function will take two parameters : First, the name of the (.json) file you want to open, then, the file you want to create (.tsv)

Step 1: make sure that the .json file specify exist and is valid.
Step 2: extract the json array elements content.
Step 3: create a file with .tsv extension
Step 4: add the content and respect the .tsv format

(every line is separated by '\n')
(every column is separated by a tab : '\t')

If the .json file doesn't exist or is not valid :

Your program will return 0
Else, you programm will return 1

You can test that your .tsv file is valid by open it on google drive
Also, you can copy the output and paste it in a google sheet (it will automatically dispatch the data in the corrects columns.

bc_test.json is available on the Bootcamp Drive Folder

For this exercise, your goal is to make it work for this file only. If you think it's too easy, you can do the same but that can adapt for any kind of data, you will get bonus!

Requirements :

- Program name : **48_json_to_tsv.py**
- Function name : **json_to_tsv**
- Directory: **week03/ex** folder

Python Bootcamp {2019}

Kirirom Institute of Technology

EXERCISE 49

Description :

You will do EXACTLY the same function that you just did (json_to_tsv) but this time, your function will take a .tsv file and return a .json file. You must create a valid JSON array.

The function will take two parameters : First, the name of the (.tsv) file you want to open, then, the file you want to create (.json)

This file :

	A	B	C
1	title	content	difficulty
2	week01	python syntax	very easy
3	week02	python data manipulation	easy
4	week03	python files and requests	intermediate
5	week04	python class and advanced concepts	hard

Will generate :


```
[{
    "title": "week 01",
    "content": "python syntax",
    "difficulty": "very easy"
  },
  {
    "title": "week 02",
    "content": "python data manipulation",
    "difficulty": "easy"
  },
  {
    "title": "week 03",
    "content": "python files and requests",
    "difficulty": "intermediate"
  },
  {
    "title": "week 04",
    "content": "python class and advanced concepts",
    "difficulty": "hard"
  }
]
```

Python Bootcamp {2019}

bc_tsv is available on the Bootcamp Drive Folder. To make your tests, click on “Download as” and choose Tab-Separated Values (TSV)

As this exercises is much more easier that the last one, your function should work for EVERY .tsv and return a correct JSON Array.

If the .tsv file doesn’t exist or is not valid :

Your program will return 0
Else, you programm will return 1

Requirements :

- Program name : **49_tsv_to_json.py**
- Function name : **tsv_to_json**
- Directory : **week03/ex** folder

Python Bootcamp {2019}

EXERCISE 50

Description :

You will create a Class that will contains useful functions that you just created. You class will be call FileLib and will contains the following functions:

- **inspect(self)**
- **current_path(self)**
- **read(self, filename)**
- **write(self, filename, content)**
- **append(self, filename, content)**
- **remove(self, filename)**
- **create_folder(self, folder_list)**

They must work exactly as the exercises you did previously without the confirmations input. Also, don't mind about self, this is a mandatory convention for every functions created in a Class (we call them methods)

If you class work you will be able to call the methods this way:

```
FileLib().inspect()
FileLib().current_path()
FileLib().read("my_file.txt")
FileLib().write("my_file.txt", "Hello World!")
...
```

Requirements :

- Program name : **file_lib.py**
- Class name : **FileLib**
- Directory : **week03/ex** folder

- Don't use number (50_file_lib.py) for this file, else you won't be able to import it as a module.