SCALE FOR PROJECT PYTHON MODULE 02 (/PROJECTS/PYTHON-MODULE-02)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere-v2-bg.1337.ma:vogsphere/intra-uuid-0dea4919-8
```

Comments

This third module is the evolution of two original projects created by the
Paris-based student organization 42 AI.
They are named Bootcamp Python and Bootcamp Machine Learning.
active members of 42 AI re-designed both of them for the school curriculum.

Bootcamps has been developped between August 2019 and March/April 2020.
Active 42AI members organized severals sessions of 2 weeks to 42Paris students
to offer them the possibility to get famliar with Python and basics concepts
of machine learning.

The success of those sections brings the pedagogy to accept the idea to integrate
the 2 bootcamps to the curriculum (initial discussion (01-05/2019) with 42 Paris
pedago team highlighted a categorical opidea)

The transcription had been realized over the direction of Matthieu David
several 42AI members contributed to the redaction on the correction scales.
For futur corrections on the scale, please contact the 42AI association via
contact@42ai.fr or the current 42AI pedagogical supervisor.

Introduction

The Bootcamp Python and Bootcamp Machine Learning were originally created by
[42AI](https://github.com/42-AI) active members and were adapted to 'piscine'
format for the school 42 curriculum.
For any issue or suggestion: [42Paris](https://github.com/42-AI/bootcamp_python/issues) and
[42AI](https://github.com/42-AI/bootcamp_machine-learning/issues).

As usual, you have to observe the following courtesy rules:

- Remain polite, courteous, respectful, and constructive throughout the evaluation process. The well-being of the community depends on it.

- Identify with the evaluated person or group the eventual dysfunctions of the assignment. Take the time to discuss and debate the problems you may have identified.

- You must consider that there might be some differences in the understanding of and approach to project instructions, and the scope of its functionalities, between you and your peers. Always remain open-minded and grade them as fairly as possible. The pedagogy is valid only and only if peer-evaluation is conducted seriously.

The goal of this module is to get started with the Python language.
You will study map, reduce, filter, args and kwargs ...

Disclaimer

The serie of modules started to be produce at the time of the release of Python 3.7. Students are free to use later version of Python as long as they verified the producted code complies with all the aspects precised in the subjects.
As a consequence we recommend to students to perform the modules with the the Python version 3.7 (but this is just an advice).
Version can be checked with the command ```python -V```.

Guidelines

General rules
- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository does belong to the student. Ensure that the work is the one expected for the corrected exercise and don't forget to verify that the command "git clone" is run in an empty folder.

- Check carefully that no malicious aliases were used to make you evaluate files that are not from the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for them to read the entire subject prior to starting the defense.

- Use the flags available on this scale to signal an empty repository,

non-functioning program, a Norm error (specified next in general rules),
cheating, and so forth.
In these cases, the grading is over and the final grade is 0,
or -42 in case of cheating. However, except the exception of cheating, you
are encouraged to continue to discuss your work even if the later is in
progress in order to identify any issues that may have caused the project
failure and avoid repeating the same mistake in the future.

- Use the appropriate flag.

- Remember that for the duration of the defense, no other unexpected,
premature, or uncontrolled termination of the program, else the final
grade is 0.

- You should never have to edit any file except the configuration file if
the latter exists. If you want to edit a file, take the time to explain why
with the evaluated student and make sure both of you agree on this.

- The Norm: The PEP 8 standards is not mandatory, but recommended.

- The function eval is never allowed.

- Your exercises are going to be evaluated by other students,
make sure that your variable names and function names are appropriate and civil.

- Copiable code is included in the code_blocks.md in attachments.
(block xx.xx.xx) is an indication that the block of code with the
same id on top should be used

Attachments

code_blocks.md (/uploads/document/document/9885/code_blocks.md)

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/55341/en.subject.pdf)

good.csv (/uploads/document/document/9886/good.csv)

bad.csv (/uploads/document/document/9887/bad.csv)

Exercise 00: Map, filter, reduce

In the error management question, function refers to an object function (a lambda function as instance) and iter refers to an iterable object. As mentioned in the subject, we do not expect the identical exception messages to be exactly the same than those of the functions `map`, `filter` and `reduce`. The goal of the exercise is to work on the built-in functions `map`, `filter` and `reduce`.

Basic tests

Perform some basic test such as the following:

- (block 02.00.00): you should get `[]`.
- (block 02.00.01) you should get `[3]`.
- (block 02.00.02): you should get `[1, 4, 9, 16, 25]`.
- (block 02.00.03): you should get `[]`.
- (block 02.00.04): you should get `1`.
- (block 02.00.05): you should get `24`. Feel free to realize few more tests.

☑ Yes ☒ No

Exercise 01: args and kwargs

The goal of the exercise is to discover and manipulate `*args` and `**kwargs`.

Basic tests

- With None as unique parameter: (block 02.01.00)
- With a function as argument and a function as keyword argument: (block 02.01.01)
- With a kwarg named var_0: (block 02.01.02)

☑ Yes ☒ No

Exercise 02: the logger

The goal of the exercise is to discover the decorator in Python and work with a very current one: "@log".

Basic tests

Verify log decorator is correctly implemented:

- It must be define in the same file.
- The definition of log takes only one parameter which is supposed to be a function.
- name is obtained via the function received as parameter (function.**name**).
- username is obtained via the environment variable.

launch the logger

Verify the machine.log contains:
(<login>)Running: <Instruction> [exec-time = <XXX> <ms\s>]

☑ Yes ☒ No

Exercise 03: Json issues

Implementation

Verify the different methods are implemented:

- **init**(self, filename=None, sep=',', header=False, skip_top=0, skip_bottom=0)
- **enter**(self)
- **exit**(self, type, value, traceback)
- getdata(self)
- getheader(self)

Ask the defendee to explain the **func** functions to you.

☑ Yes ☒ No

Basic tests

Perform the following tests. They should work and give correct results.
Create a file called main.py and copy the following code in that file.

(block 02.03.00)

Put the files bad.csv and good.csv in the current folder then run in terminal:
(block 02.03.01)
The appearance of the output may vary slightly but the content and number of lines must be the same as the output above.
(block 02.03.02)

(block 02.03.03)
Additional error messages may print no python messages containing `Traceback` (these indicate a crash of the python interpreter even though the python console keeps running in the terminal)

☑ Yes ☒ No

Exercise 04 - MiniPack

The goal of the exercise is to learn how to build a package and a distribution in Python.

turn-in files

A python package has to be constituted of the following files according to
the file structure:
ex02/
├── LICENSE.txt (mandatory for the exercise, recommended otherwise)
├── pyproject.toml (recommended)
├── README.md (mandatory for the exercise, recommended otherwise)
├── setup.cfg (mandatory or setup.py)
├── setup.py (mandatory or setup.cfg)

```
├── src/
|  └──────my_minipack/
|  |  ├──── init.py
|  |  ├──── logger.py
|  |  └──── progressbar.py
└──── tests/ (not asked here, recommended otherwise)
```

First, check the different files in the directory. License, README,
setup.py and setup.cfg files must be present and a directory containing the sources.
In the sources directory you should have 3 python files (*names may not be exactly the same*):

- **init**.py
- logger.py
- progressbar.py

☑ Yes ☒ No

---

build script

Begin by running:

(block 02.04.00)

if any files are found delete them (make sure you are in the ex04 folder)

(block 02.04.01)

Now the defendee should tell you how to install his package in a few lines.

(block 02.04.02)

you should now be able to import the package !

(block 02.04.03)

☑ Yes ☒ No

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok

Empty work Cheat Crash Incomplete group Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation