

≡ 🔍 (https://profile.intra.42.fr/searches)

(https://profile.intra.42.fr)

SCALE FOR PROJECT PYTHON MODULE 01 (/PROJECTS/PYTHON-MODULE-01)

You should evaluate 1 student in this team



Comments

This second module is the evolution of two original projects created by the Paris-based student association 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 opposition/refusal to this idea)

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 objects, classes, inheritance, built-in functions, magic methods, generator ...

Disclaimer

The series of modules started to be produced at the time of the release of Python 3.7. Students are free to use later versions of Python as long as they verified the produced code complies with all the aspects specified in the subjects.

As a consequence we recommend to students to perform the modules with 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` (https://cdn.intra.42.fr/document/document/11895/code_blocks.md)

 `subject.pdf` (<https://cdn.intra.42.fr/pdf/pdf/64625/en.subject.pdf>)

 `banking_test2.py` (https://cdn.intra.42.fr/document/document/11896/banking_test2.py)

 `banking_test3.py` (https://cdn.intra.42.fr/document/document/11897/banking_test3.py)

 `banking_test1.py` (https://cdn.intra.42.fr/document/document/11898/banking_test1.py)

Exercise 00 - The Book

The goal of the exercise is to get you familiar with the notions of classes and the manipulation of the objects related to those classes.

Error management and basic tests

There are 2 classes, Book in book.py and Recipe in recipe.py.

A Book object stores and manages Recipe.

With the help of the test.py file, carry out at least the following tests to try to stress the error management:

before anything, run (block 01.00.00)

- Recipe: (block 01.00.01) Should print an error because cooking_lvl < 1

(block 01.00.02)

Should print an error because cooking_lvl is not an int

OR

cooking_lvl should be converted to an int somewhere

(block 01.00.03)

Should print an error because empty ingredients

(block 01.00.04)

- Book:
 - After the initialization of a correct Book object, verify: (block 01.00.05)
 - get_recipe_by_name() tests: (block 01.00.06)

Should be a different date / time than the one printed before
(block 01.00.07)

 - get_recipes_by_types() tests: (block 01.00.08)
 - With an correct type (with at least a recipe), the function must print **all** the recipes which match the type.
 - With a correct type which exists but being empty. This case must be handled properly
 - Verify also 'last_update' did not change
 - With a non existing recipe type, the function must declare an error and handle it correctly. Returning an empty array is also acceptable.

✓ Yes

✗ No

Exercise 01 - Family Tree

The goal of the exercise is to tackle the notion inheritance of class.

notion of class

The exercise shows an example of inheritance of a class from a parent class.

- Verify the implementation of the differents classes (GotCharacter and the children), parent should contains the attributes 'first_name' and 'is_alive' while the children classes must have 'family_name', 'house_words' and the 2 methods 'print_house_words' and 'die'
- Verify when creating an instance of the GotCharacter class without any value that you get an object with the attributes 'first_name' sets to None and 'is_alive' sets to True.

- Perform the same test and specify a value for the first_name and is_alive, for example: `luigi = GotCharacter("Name", True)` Verify that the GotCharacter instance is alive and it's name.
- Check the presence of the docstring `print(luigi.__doc__)`
(Some python setting can make this return None, if that's the case, go check in the code for an explanative text below `def GotCharacter():`)
- Verify that the GotCharacter instance has no attributes 'house_words' or 'family_name' (of course you should not be able to call `print_house_words()` or the `die()` neither)
- children class
 - Creates an instance of the children class and checks: `arya = Stark("Arya")`
 - Check the presence of the docstring `print(arya.__doc__)`
 - Verify all the attributes of the character
 - Verify that the methods associated to the character object
 - Use the method `die()` and verify the value of `is_alive`
 - Create a new instance with a different name, and verify it is alive and its name

☒ Yes

☐ No

Exercise 02 - The Vector

The goal of the exercise is to get you used with built-in methods, more particularly with those allowing to perform operations. Student is expected to code built-in methods for vector-vector and vector-scalar operations as rigorously as possible. Concerning the `__str__` and `__repr__` you have the opportunity to observe their behaviors through the tests.

Implementation of built-in methods

First, verify all the expected built-in methods are implemented, and then printing a vector you observe an output similar to:
(Vector [n1, ..., n2])
can note that clever implementation may use **add** method within **radd**, **sub** and **rsub**. Nevertheless it is perfectly fine if it is not the case.

☒ Yes

☐ No

built-in method `__init__`

Verify the correct implementation of the **init** method:
Create multiple instances of Vector class according to the different possibilities and verify if the values and shape are correct:

- (block 01.02.00)
Output: [1.0, 2e-3, 3.14, 5.0]

- (block 01.02.01)
Output: `[[0.0], [1.0], [2.0], [3.0]]`
- (block 01.02.02)
This is an error and should be properly handled by displaying an error message.
- (block 01.02.03)
Output: `[[10.0], [11.0]]`
- (block 01.02.04)
Should display a properly handled error or `[[3.0], [2.0]]`
- (block 01.02.05)
three behaviours are acceptable:
raise an error or print `[]` or print `[[]]`
- (block 01.02.06)
This is an error and should be properly handled by displaying an error message.

☒ Yes☐ No

built-in method `__mul__` and `__rmul__`

Verify the correct implementation of the built-in method **mul** and **rmul** by performing several tests related to '*' operator:

(block 01.02.07)

Should output: `[[0.0], [1.0], [2.0], [3.0]]`

(block 01.02.08)

Should output: `[[0.0], [4.0], [8.0], [12.0]]`

(block 01.02.09)

Should output: `[[0.0], [4.0], [8.0], [12.0]]`

(block 01.02.10)

The error should be handled, it should say raise `SomeError` in the code or print a message or return `None` or `exit()`.

☒ Yes☐ No

built-in method `__add__`, `__radd__`, `__sub__` and `__rsub__`

Verify the correct implementation of the built-in method **add**, **radd**, **sub** and **rsub** by performing several tests related to '+' and '-' operators:

(block 01.02.11)

Should output: `[[1.0], [2.0], [3.0], [4.0]]`

(block 01.02.12)

The error should be handled in some way

(block 01.02.13)

The error should be handled in some way

(block 01.02.14)

The error should be handled in some way

(block 01.02.15)

Should output: True

☒ Yes

☐ No

built-in method `__div__` and `__rdiv__`

For division perform the following tests:

- (block 01.02.16)
- (block 01.02.17)
- (block 01.02.18) An error should be displayed, and properly handled.
- (block 01.02.19) and (block 01.02.20) An error should be displayed, and properly handled
- (block 01.02.21) should raise an error or the choice of behaviour must be justifiable as multiple mathematical definitions of division by a vector exist.

☒ Yes

☐ No

Exercise 03 - Generator!

The goal of the exercise is to discover the concept of generator object in Python, through the implementation of a function 'generator' which performs a splitting operation and an ordering class operation.

Error managment and basic tests

- First, check the implementation of the function (all behaviors are implemented: 'shuffle', 'ordered' and 'unique').
- Secondly, with 'text' parameter being a string with multiple words and punctuation (for example text="This is a simple string for a basic test. Very simple."), runs the tests:

(block 01.03.00)

For the given example the tests give (with line break between each strings):

- First test: 'This' 'is' 'a' 'simple' 'string' 'for' 'a' 'basic' 'test.' 'Very' 'simple.'
- Second test: 'This is a simple string for a basic test' ' Very simple' "
- Third test: 'Th' 's' ' 's a' 'mple str' 'ng for a bas' 'c test. Very s' 'mple.'
- Fourth test: 'This is a ' 'mple strng for a ba' 'c test. Very ' 'mple.'
- Finally, with 'text' parameter being a non empty string you must test 'option' parameter. Check the value 'ordered' gives a alphanumerical ordering, 'unique' gives a set of the words (no identical strings) and 'shuffle' randomly arrange the words. Spend a particular attention on the last value, 2 runs with 'shuffle' should not give the same arrangement (th opposite is highly unlikely). Verify also there is no missing words.

☒ Yes☐ No

Exercise 04 - Working with lists

The goal of the exercise is to discover 2 useful methods for lists, tuples, dictionaries (iterable class objects more generally) named `zip` and `enumerate`.

Error management and basic tests

- First, check that the implementation of the class and methods.
You must find 'zip' in the method 'zip_evaluate' and 'enumerate' in 'enumerate_evaluate' and no while loop. It is a normal implementation to find 'enumerate(zip(coefs, words))' so do not be surprised.
- Then, test a few cases with lists length 1 and 2 to check the validity of the implementation and check no term of lists are missing in the sum of products (especially the last one).

☒ Yes☐ No

Exercise 05 - Bank account

The goals of this exercise is to teach you new built-in functions and get a deeper understanding about class manipulation and to be aware of possibility to modify instanced objects. In this exercise you learn how to modify or add attributes to an object.

Basic tests

You have to check the classes implementation, meaning that you have to check the presence of:

- the class `Account` and `Bank` in the file `bank.py`
- the methods 'init', 'transfer' and the attribute 'ID_COUNT' in the class `Account`.
- the methods 'init', 'add', 'transfer' and 'fix_account' in the class `Bank`. Student may have implement extra methods to manage the security aspect, thus it is okay to have more than the methods mentionned.

☒ Yes☐ No

For the following accounts ask the defendee to run his corruption detection function

(block 01.05.00)

- The following account is corrupted because it has an attribute that starts with b:
(block 01.05.01)
This should fix john OR return a fixed copy of john

- The following account is corrupted because it has an even number of attributes:
(block 01.05.02)
- The following account is corrupted because it has no attribute "value":
(block 01.05.03)
- We will now test a bank transfer
(block 01.05.04)
Should output 1000.0 then True then 1500.0 . The transfer worked .
(block 01.05.05)
Should output False then 1500.0 OR Raise an Error.

Jhon's balance MUST not change.

☒ 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

Rules of procedure (<https://profile.intra.42.fr/legal/terms/4>)

Declaration on the use of cookies (<https://profile.intra.42.fr/legal/terms/2>)

Privacy policy (<https://profile.intra.42.fr/legal/terms/5>)

General term of use of the site (<https://profile.intra.42.fr/legal/terms/6>)

Terms of use for video surveillance (<https://profile.intra.42.fr/legal/terms/1>)

Legal notices (<https://profile.intra.42.fr/legal/terms/3>)