(https://profile.intra.42.fr/searches) **suham**

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

SCALE FOR PROJECT PYTHON - 0 - STARTING (/PROJECTS/PYTHON-0-STARTING)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.42seoul.kr:vogsphere/intra-uuid-56a5f147-193c
```

Introduction

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

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

- You must consider that there might be some difference in how your
peers might have understood the project's instructions and the
scope of its functionalities. Always keep an open mind and grade
him/her as honestly as possible. The pedagogy is valid only and
only if peer-evaluation is conducted seriously.

Guidelines

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

- Double-check that the GiT repository belongs to the student
or the group. Ensure that the work is for the relevant project
and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you
and make you evaluate something other than the content of 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 this student to read the
entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository,
non-functioning program, 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.

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

- 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.

- Lib imports must be explicit, for example importing "from pandas import *"
is not allowed, you must put 0 to the exercise and continue the evaluation.

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

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/116334/en.subject.pdf)

Mandatory Part

Error Management

Carry out AT LEAST the following tests to try to stress the error
management

- The repository isn't empty.
- No cheating.
- No forbidden function/library.
- The executable is named as expected.
- Your lib imports must be explicit, for example you must "import numpy as np". (Importing "from pandas import *" is not allowed, and you will get 0 on the exercise.)
- If an exercise is wrong, go to the next one.

✓ Yes ✕ No

ex00 First python script.

- Read the file Hello.py and make sure that the displayed items are indeed the objects and not just strings otherwise you can mark the exercise as wrong and move on to the next part of the evaluation.
- It can happen that the program displays: {'Paris!', 'Hello'},
  this is not wrong, you can ask the evaluated why this happens.

The exercise is incorrect if one of the requirements is not perfectly met.

✓ Yes ✕ No

ex01 First use of package

The time must change, if the student has printed a fixed time,
put 0 and continue the correction.

```
$>python format_ft_time.py | cat -e
Seconds since January 1, 1970: 1,666,355,857.3622 or 1.67e+09 in sc
ientific notation$
Oct 21 2022$
$>
```

✓ Yes ✕ No

ex02 First function python

The function must print the type of object sent as argument and return 42.

Your tester.py:

```
from find_ft_type import all_thing_is_obj

ft_list  = ["Hello", "tata!"]
ft_tuple = ("Hello", "toto!")
ft_set   = {"Hello", "tutu!"}
ft_dict  = {"Hello" : "titi!"}

all_thing_is_obj(ft_list)
all_thing_is_obj(ft_tuple)
all_thing_is_obj(ft_set)
all_thing_is_obj(ft_dict)
all_thing_is_obj("John")
print(all_thing_is_obj(1.5))
```

expected output:

```
$> python tester.py
List : <class 'list'>$
Tuple : <class 'tuple'>$
Set : <class 'set'>$
Dict : <class 'dict'>$
John is in the kitchen : <class 'str'>$
Type not Found$
42$
```

The exercise is incorrect if one of the requirements is not perfectly met.

✓ Yes ✕ No

ex03 NULL not found

The function must print the type of NULL and return 0 if it goes well and 1 in case of error.
Your tester.py:

```
from NULL_not_found import NULL_not_found

Nothing = None
Garlic  = float("NaN")
Fake    = False
Zero    = 0
Empty   = ""

NULL_not_found(Nothing)
NULL_not_found(Garlic)
NULL_not_found(Zero)
print(NULL_not_found(Empty))
NULL_not_found(Fake)
print(NULL_not_found("4.2"))
```

expected output:

```
$>python tester.py | cat -e
Nothing: None <class 'NoneType'>$
Cheese: nan <class 'float'>$
Zero: 0 <class 'int'>$
Empty:  <class 'str'>$
0$
Fake: False <class 'bool'>$
Type not Found$
1$
$>
```

The exercise is incorrect if one of the requirements is not perfectly met.

✓ Yes ✗ No

ex04 The Odd and the Even

Carry out at least the following tests to try to stress the error management:

Without argument: The program must either do nothing or display the usage.
Test it with a valid input: odd number, even number, negative number.
With more than one argument : The program must report an error and quit
With non integer parameters : The program must report an error and quit, test it with at least a string and a float.

Look in the code to see if the student has used the AssertionError if not it is 0 at the exercise

Expected output:

```
$> python whatis.py 28
I'm Even.
$>
$> python whatis.py -7
I'm Odd.
$>
$> python whatis.py
$>
$> python whatis.py 0
I'm Even.
$>
$> python whatis.py "Hola!"
AssertionError: argument is not an integer
$>
$> python whatis.py 42 42
AssertionError: more than one argument are provided
$>
```

✓ Yes ✗ No

clean coding

New rules

From now on you must follow these additional rules

- There is no global variable.
- Norminette shows no errors. (pip install flake8, alias norminette=flake8, use flag Norme)
- All your functions must have a documentation (__doc__)
- No code in the global scope. Use functions!
- Each program must have its main and not be a simple script

```
if __name__ == "__main__":
  # your tests and your error handling
```

✓ Yes ✗ No

ex05 First standalone program python

Test the program with strings directly in the argument or in the program prompt
by running the program without arguments.
If more than one argument is provided to the program, print an AssertionError.

Expected output:

```
$>python building.py "Python 3.0, released in 2008, was a major rev
ision that is not completely backward-compatible with earlier versi
ons. Python 2 was discontinued with version 2.7.18 in 2020."
The text contains 171 characters:
2 upper letters
121 lower letters
8 punctuation marks
25 spaces
15 digit
```

Expected output: (the carriage return counts as a space, if you don't want to return one use ctrl + D)

```
$>python building.py
What is the text to count?
Hello World!
The text contains 13 characters:
2 upper letters
8 lower letters
1 punctuation marks
2 spaces
0 digits
```

Expected output: (error handling can be different)

```
$>python building.py "Hello" "World!"
AssertionError: more than one argument are provided
```

✓ Yes ✕ No

ex06 Part 1

Examine the source code of ft_filter.py and make sure it follows the following restrictions:

- The function have a docstring.
- The function use a list comprehension.
- The function dont use the original built-in filter

Your tester.py:

```
from ft_filter import ft_filter
original = filter.__doc__
copy  = ft_filter.__doc__

print(copy) # output: docstring
print(original == copy) # output: True
```

✓ Yes ✗ No

ex06 Part 2

Examine the source code of program. Make sure it follows the following restrictions:

- The program use a list comprehension and a lambda.
- The program use AssertionError for the error management

Make your own test without using Punctuation or invisible character,
the goal of this exercise being the comprehension list and lambda not a perfect error management.

Expected outputs:

```
python filterwords.py 'A robot may not injure a human being or thro
ugh inaction
allow a human being to come to harm' 5

['injure', 'through', 'inaction']
```

✓ Yes ✗ No

ex07 Dictionaries SoS

Examine the source code of program. Make sure it follows the following restrictions:

- The program use a dictionary.
- The program use AssertionError for the error management

Carry out at least the following tests to try to stress the error management:

- Test the program without arguments, with more 1 arguments
- Test the program with special character '$', '@' ...

Expected outputs:

```
python sos.py 'SOS' | cat -e
... --- ...$
```

Make your own tests, you can check the good transformation with this site for example:
https://morsecode.world/international/translator.html
(https://morsecode.world/international/translator.html)

Yes No

ex08 Loading ...

Compare the ft_tqdm function with the real one, they should be comparable,
but don't be too demanding, 1 or 2 pixels of difference are acceptable.

Your tester.py:

```
from time import sleep
from tqdm import tqdm
from Loading import ft_tqdm

for elem in ft_tqdm(range(333)):
  sleep(0.005)
print()
for elem in tqdm(range(333)):
  sleep(0.005)
print()
```

Expected output:

```
$> python tester.py
100%|[============================================================
==>]| 333/333
100%|


The "[00:01<00:00, 191.61it/s]" is not necessary,
in this exercise we want to learn the yield operator.
```

Yes No

ex09 My first package creation

write the following command and check the correct installation of the
package

```
pip install ./dist/ft\_package-0.0.1.tar.gz
pip install ./dist/ft\_package-0.0.1-py3-none-any.whl

pip show -v ft\_package
```

Write a test script to import the package and test it

```
from ft_package import count_in_list

print(count_in_list(["toto", "tata", "toto"], "toto")) # output: 2
print(count_in_list(["toto", "tata", "toto"], "tutu")) # output: 0
```

✓ Yes ✕ No

Ratings

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

✔ Ok ★ Outstanding project

Empty work Incomplete work Norme Cheat Crash

⚠ Concerning situation ⊘ Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation