SCALE FOR PROJECT PYTHON MODULE 00 (/PROJECTS/PYTHON-MODULE-00)

You should evaluate 1 student in this team

★

Git repository

`git@vogsphere-v2.1337.ma:vogsphere/intra-uuid-92ecb943-657d`

Comments

This first 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 20119 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_staff_pedago](https://42born2code.slack.com/archives/C7NF60E0Z) 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 basic setup, variables, data types, functions, ...

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: You will follow the PEP 8 standards.

- 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.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/33904/en.subject.pdf)

Exercise 00 - $PATH

Python environment

The goal of the exercise is to understand how to set up a python
environment and install some packages.

- Verify that the list of packages and its versions correspond to those of the installed miniconda environment.
To get the list of the packages and its versions run the following command: ```conda list```.
Note that the version of a package and the version of Python evolved, thus do not refer to the lastest version
on conda website and pypi.org (or any other python package library).
- Verify that the content of the file answers.txt corresponds to the return of the command ```pip search tesseract -i https://pypi.org/pypi```
- Verify that the content of the file answers.txt corresponds to the return of the command ```pip freeze > requirements.txt```

Yes No

Exercise 01 - Rev Alpha

correction

The goal of the exericse is to manipulate functions relative a string
object.

The program reverses the order of a string and the case, and then print it.
If multiple arguments are passed they must be joined together with a space.

- 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.
- With a valid string, i.e. :
```python exec.py "Hello" -> "OLLEh"```
- With a string containing non alhabetical characters, i.e. :
```python exec.py "L337 5P3AK!" -> "!ka3p5 733l"```
- With multiple arguments, i.e. :
```python exec.py "L337" "5P3AK!" -> "!ka3p5 733l"```
- With multiple arguments, with some being empty strings. Verify the presence of the added space between each argument
```python exec.py "L337" "" "5P3AK!" -> "!ka3p5 733l"```

Yes No

Exercise 02 - The Odd, the Even and the Zero

correction

The goal of the exercise is to manipulate integers with Python and
perform some conditional tests based on modulo.

The program checks if a number is odd, even or zero.

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, of each type : 0, an odd number, and an even 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

Yes No

Exercise 03 - Functional file

correction

The goal of the exercise is to work with built-in string functions and
give the possibility to work with formatted strings.

Carry out at least the following tests to try to stress the error
management:
- Without argument:
The program must take the input for the user and process it.
- Test it with a couple of valid inputs
- Test it with an empty string
- With more than one argument : The function must report an error and quit

Yes No

Exercise 04 - Elementary

correction

The goal of the exercise is to manipulate operator for numbers and
allows student to work with string formatting.

The program prints the results of the four elementary mathematical
operations of arithmetic (addition, subtraction, multiplication,
division) plus the modulo operation.

Carry out at least the following tests to try to stress the error
management:
- Without argument, one argument, and three, or more, arguments :
The program must report a descriptive error and must display an usage
- Test it with a couple of valid inputs : positive and negative integers
- Test it with the second parameter being 0 :
The program must normally display the sum, difference and product,
but must write an error as a result of the modulo and division operation.
- Test it with invalid inputs : like strings or floats
The program must report a descriptive error and must display an usage

Yes No

Exercise 05 - The right format

correction

The goal of the exercise is to discover tuple and dictionary object and
also the formatted strings.

These programs prints a formatted string from different data types.

Carry out at least the following tests to try to stress the error
management:
- For each kata, verify the exactitude of the formating with the subject
- kata00 : with tuples of different sizes, as well as an empty one
- kata01 : with dictionaries of different sizes, as well as an empty one
- kata02 : with a tuples of 5 numbers (hour, minute, year, month, day)
hour, minute, month and day must have a minimum width of 2 and fill character set to '0'
year must have a minimum width of 4 and fill character set to '0'
- kata03 : with strings of varying sizes (below 43 characters), as well as one empty string
They must be printed without ending '\n', and must output exactly 42 characters.
- kata04 : with a tuples of 5 numbers.
- 1st and 2nd elements must have a minimum width of 2 and fill character set to '0'
- 3rd element must be displayed with a precision of 2
- 4th and 5th elements must be displayed with scientific notation and a precision of 2

☑ Yes ☒ No

Exercise 06 - A recipe

correction

The goal of the exercise is to learn to work with a dictionnary.

This program interactively manages cookbook recipes.

Carry out at least the following tests to try to stress the error
management:
- From the menu from which to select an action :
Try a non existing option : The program must display a descriptive error and let you input again
- Print the cookbook
- Print one recipe from the cookbook
- Ask to print a non existing recipe from the cookbook
The program must display a descriptive error and let you input again
- Create a new recipe, and then print it
- Delete all recipes from the cookbook, the ask to print the cookbook
This case must be properly managed by the student program, the program should not exit unexpectedly

Yes No

Exercise 07 - Shorter, faster, pythonest

correction

The goal of the exercise is to discover and work with list comprehension.
This program removes all the words in a string that are shorter or equal than n letters, and returns the filtered list with no punctuation.

You must make sure the program use list comprehensions to solve this exercise
In Python list comprehensions take the form of [operation(n) for n in array if condition(n)]

Carry out at least the following tests to try to stress the error
management:
- Test the program without arguments, a wrong number of arguments and wrong argument's types :
The program must declare an error and exit
- Test the program with an empty string
- Test the program with only one word
- Test the program with a string of multiple words
- Test the program with a string of multiple words and ponctuation
- Test the program with a string of multiple words and punctuation within words
- Test the program with valid arguments to output an empty array

Yes No

Exercise 08 - S.O.S

correction

The program encodes a string to Morse code, and then prints it on stdout.
All alphanumeric characters must be recognized. If another character is
input, the program must declare an error and quit.

Carry out at least the following tests to try to stress the error
management:
- Test the program without arguments :
The program must do nothing
- Test the program with "SOS"
output should be : "... --- ..."
- Test the program with "sos"
output should be : "... --- ..."
- Test the program with "69"
output should be : "-.... ----."

- Test the program with the 2 arguments "a1 B2" "c3D4e5F6"
output should be : ".- .---- / -... ..--- / -.-. ...-- -.. ....- . ...-. -...."
- Test the program with the 2 arguments "He||o"
The program must print "ERROR" (because of the non alphanumeric character)

☑ Yes ☒ No

Exercise 09 - Secret number

correction

The program is an interactive guessing game. It will ask the user to guess a number between 1 and 99. The program will tell the user if their input is too high or too low. The game ends when the user finds out the secret number or types 'exit'.

Carry out at least the following tests to try to stress the error management:
- Test the program by doing a normal game :
The program must declare your victory and correctly tell you how many attempts you took
- Test the program by launching a normal game, and then enter 'exit' to make the program print 'Goodbye!' then exit.
- Test the program by doing a game where you enter the following bad inputs:
- 'hello', 'a', '[0]' : The program must not quit, and it must allow you to enter a new input.
- '0', '100', '-42' : The program must not quit, every input must count as an attempt (even with a wrong one).
- '3.14', '1e7' : The program must not quit, every input must count as an attempt (even with a wrong one).
- Finish the game, and verify that the announced number of attempts does count the invalid inputs

☑ Yes ☒ No

Exercise 10 - Loading bar!

correction

The goal of the exercise is to softly start to discover the generator in Python.

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

management:
In a test.py file, in which you will import the user function:
```
from loading import ft_progress

for elem in ft_progress(X):
ret += (elem + 3) % 5
sleep(0.01)
print()
print(ret)
```
You will test for the following values for X:
Each of this inputs is valid and should not produce any error.
- X = range(100)
- X = range(100, 200)
- X = range(1)
- X = range(4)
- X = range(0, -100, -1)

☑ Yes ☒ No

Ratings

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

✔ Ok

Empty work | Norme | Cheat | Crash | Incomplete group | Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation