SCALE FOR PROJECT PYTHON MODULE 00

You should evaluate 1 student in this team

Comments

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

Bootcamps has been developpped between August 2019 and (more) April 2020.
Active 42IA members organized several sessions of 2 weeks to 10 Paris students
to offer them the possibility to get familiar with Python and basics concepts
of machine learning.

The success of those sessions brings the pedagogy to accept the client to integrate
the 2 bootcamps in the curriculum (initial discussion 20/05/2019 with 42 Paris
pedagogy 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 fast 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 (42piscine_staff_pedago@https://42born2code.slack.com/archives/C7PMN4R5Z) 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 subject. 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
the project. Be open-minded and grade your peer with the benefit of the doubt
and guide 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 aimed with the Python language.
You will study basic setup, variables, data types, functions, ...

Disclaimer

The aims of modules aimed 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
reflect the production code compliac with all the aspects precised in the
subjects.
The pedagogues are recommended 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 evaluated students have reviewed the possible scripts used
to facilitate the peer-evaluation.

- 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 File to name differently/not in general (wrong),
cheating, and so forth.
With the flags, the grade is zero 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 latter 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, alter the final
grade to 0.

- You should never have to edit any File except the configuration file if
the latter exists. If you wish 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.rr

- Copyable code is included in the code_blocks.txt attachments.
[block.sa.ex.ex] is an indication that the block of code with the
same id in tag should be used

Attachments

code_blocks.md

subject.pdf

Exercise 00 - $PATH

Python environment

correction:
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 python environment. To get the list of the packages and its versions run the following command [block.00.01.00] Note that the version of a package and the version of Python evolved, thus do not hesitate to the latest version on conda website and pypi.org (or any other python package library).
- Verify that the content of the file anaconda.txt corresponds to the return of the command [block.00.00.01]

Yes No

Exercise 01 - Rev Alpha

correction:
The goal of the exercise is to manipulate/sections 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. [block.00.01.00.10].
  - With a string containing non-alphabetical characters, i.e. [block.00.01.01]
  - With multiple arguments, i.e. [block.00.01.02]
  - With multiple arguments, with some being empty strings. Verify the presence of the added space between each argument i.e. [block.00.01.03]

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.
- With one integer argument: The program must print the appropriate message
- 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 Such with a boolean 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 valid inputs: positive and negative integers. The program must correctly display the sum, difference and product, but must write an error as 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 tuple, verify the exactitude of the formating with the subject.
  - tuple01 : with tuples of different sizes, as well as an empty one
  - tuple02 : with dictionaries of different sizes, as well as an empty one
  - tuple03 : with tuples of different sizes, as well as an empty one. The hour, minute, month and day must have a minimum width of 2 and the character sent to '0' year must have a minimum width of 4 and the character sent to '0'
  - tuple04 : with tuples of varying sizes (before v3.7 mentioned), as well as one empty string (they must be printed without making "/n", and must output exactly 42 characters.
  - tuple05 : with tuples of different sizes
  - 1st and 2nd elements must have a minimum width of 2 and the character sent to '0'
  - 3rd elements must be displayed with a precision of 3
  - 4th and 5th elements must be displayed with correctly rotation and a precision of 2

Yes No

Exercise 06 - A recipe

correction:
The goal of the exercise is to learn to work with a dictionary.

This program interactively manages cookbook recipes.

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

- Have the menu from which to select an option - 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
- Delete one 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 exit 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.

Filter the keys of the words in a string that are shorter or equal than a letters, and returns the filtered list with
its provisions.

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

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 disclose 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 punctuation
- 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 in Morse code, and then prints it on stdout.
All alphanumeric characters must be recognized. If another character is
input, the program must disclose 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 "42" output should be "...._...._..."
- Test the program with the 2 arguments "42" "42" "42" output should be "...._...._...._...._...._...."
- Test the program with "Hey Jude" 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
enter a number between 1 and 99. The program will then 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 disclose 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', '0', '100' : The program must not exit, and it must allow you to enter a new input
  - '-1', '101', '42' : The program must not exit, every input must counter as an attempt (even with a wrong one)
  - '1', '42' : The program must not exit, 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 safely 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:
[block.00.10.00]
You will test for the following values for X:
(The output should be valid and should not produce any error:
- X = range(100)
- X = range(100, 200)
- X = range(1)
- X = range(42)
- X = range(0) - 100) - 0)

Yes

Ratings

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

✔ OK

Empty work · Cheat · Crash · Incomplete group · Forbidden function