SCALE FOR PROJECT PYTHON MODULE 04 (/PROJECTS/PYTHON-MODULE-04)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere-v2.1337.ma:vogsphere/intra-uuid-f3399091-2c3b
```

Comments

This fifth 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 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_pedagogy](https://42born2code.slack.com/archives/C7NF60E0Z)
and
[42AI](https://github.com/42-AI/bootcamp_python/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 library Pandas and the manipulation of dataframes.

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/42656/en.subject.pdf)

athlete_events.csv (/uploads/document/document/7265/athlete_events.csv)

Exercise 00 - FileLoader

The goal of this exercise is to create a Fileloader class containing a load and a display method.

Error management

No specific error management is asked, but the functions should not crash or leave in an unexpected way. Thus test the following cases:
- passing a non existing file to load method,
- passing an argument different than a string type to load method,
- passing a non pandas.DataFrame to display method as first parameter,
- passing a non integer parameter to display method as second parameter.

Yes No

Load and display methods tests

Check if load method works correctly with a valid path:
- returns a pandas.DataFrame object
- displays the number of rows and columns loaded in the DataFrame,
for the 'athlete_events.csv' 271116 rows and 14 columns.

☑ Yes ☒ No

Exercise 01 - YoungestFellah

The goal of this exercise is to create a function that will return a dictionary containing the age of the youngest woman and the youngest man who took part in the Olympics a given year.

Error management

No specific error management is expected, but the function should not
crash or leave in an unexpected way. Test 'youngestfellah' by performing at least the following cases:
- Passing a non existent year to youngestfellah for year parameter.
- Passing a non pandas.DataFrame parameter for data parameter.

☑ Yes ☒ No

Basic tests

Check the results of the following cases:
- "youngestfellah(data, 1992)", output is: "{'f': 12.0, 'm': 11.0}"
- "youngestfellah(data, 2004)", output is: "{'f': 13.0, 'm': 14.0}"
- "youngestfellah(data, 2010)", output is: "{'f': 15.0, 'm': 15.0}"
- "youngestfellah(data, 2003)", output is: "{'f': nan, 'm': nan}"

Odd years, years before 1952 and after 2008 must return nan for both gender or a similar result.
If something does not match, the exercise is failed.

☑ Yes ☒ No

Exercise 02: ProportionBySport

The goal of this exercise is to create a function displaying the proportion of participants who played a given sport, among the participants of a given genders.

Error managment

No specific error management is expected, but the function should not
crash or leave in an unexpected way. Thus test 'proportionBySport':
- by giving a non pandas.DataFrame for data parameter
- by giving a non integer type parameter for year parameter
- by giving a non string type parameter for sport parameter
- by giving a non string type parameter for gender parameter
- by giving a non existent year (a year in the past or in the futur)
- by giving a sport not in the dataframe
- by giving a string different from 'M' and 'F' for the gender.

Yes No

Basic tests

Verify the results of the following cases:
- "proportionBySport(data, 2004, 'Tennis', 'F')" , output is "0.01935634328358209"
- "proportionBySport(data, 2008, 'Hockey', 'F'))", output is "0.04149467738431458"
- "proportionbysport(data, 1964, 'Biathlon', 'M'))", output is "0.009539842873176206"
If something does not match, the exercise is failed.

Yes No

Exercise 03: HowManyMedals

The goal of this exercise is to code a function that will return a dictionary of dictionaries giving the number and type of medals for each year during which the participant won medals.

Error managment

No specific error management is expected, but the function should not
crash or leave in an unexpected way. Thus test 'howManyMedals':
- by giving a non expected type argument for data or participant parameter
- by giving a empty pandas.DataFrame for data parameter
- by giving a random string for the participant
- by giving a parameter different than a string.

Yes No

Basic tests

Verify the results of the following cases:
- "howManyMedals(data, 'Gary Abraham')", the output is: "{1976: {'G': 0, 'S': 0, 'B': 0}, 1980: {'G': 0, 'S': 0, 'B': 1}}"
- "howManyMedals(data, 'Yekaterina Konstantinovna Abramova')", the output is "{2006: {'G': 0, 'S': 0, 'B': 1},

2010: {'G': 0, 'S': 0, 'B': 0}}"
- "howManyMedals(data, 'Kristin Otto')", the output is: "{1988: {'G': 6, 'S': 0, 'B': 1}}"
If something does not match, the exercise is failed.

☑ Yes ☒ No

Exercise 04 - SpatioTemporalData

The goal of this exercise is to write a class called SpatioTemporalData that takes a dataset (pandas.DataFrame) as argument in its constructor and implements two methods.

Error management

No specific error management is expected, but the function should not
crash or leave in an unexpected way. Thus test 'where' and 'when':
- by giving to 'where' a non integer type parameter,
- by giving to 'when' a non string parameter,
- by giving to 'where' a year not present in the dataset,
- by giving to 'when' a city (string object) not present in the dataset.

☑ Yes ☒ No

Basic tests

Check the result of the following cases:
-" print(sp.where(2000))", output is: "['Sydney']"
- "print(sp.where(1980))", output is: "['Lake Placid', 'Moskva']"
- "print(sp.when('London'))", output is: "[2012, 1948, 1908]"
You should try other locations and years.
If something does not match, the exercise is failed.

☑ Yes ☒ No

Exercise 05 - HowManyMedalsByCountry

The goal of this exercise is to write a function that returns a dictionary of dictionaries giving the number and type of medal for each competition where the country delegation earned medals.

Error management

No specific error management is expected, but the function should not
crash or leave in an unexpected way. Thus test 'howManyMedalsByCountry':
- by giving to 'data' a non pandas.DataFrame type parameter
- by giving to 'country_name' a non string parameter
- by giving to 'country_name' a country name not present into the dataset.

Yes No

Basic tests

Print the result of howManyMedalsByCountry functions calls with various countries
and check that the format respects the following.
"{1952: {'G': 0, 'S': 0, 'B': 0}, 1956: {'G': 0, 'S': 0, 'B': 0}, 1960: {'G': 0, 'S': 0, 'B': 0}, 1964: {'G': 0, 'S': 0, 'B': 1}}"
Try the functions with various countries and check for inconsistencies.

Yes No

Exercise 06 - MyPlotLib

The goal the exercise is to introduce plotting methods among the different libraries Pandas, Matplotlib, Seaborn or Scipy

Error management

No specific output or behavior is expected if a parameters are wrongs or
if non expected parameters are given, nethertheless a basic management
is expected and those cases should be handled properly:
- giving a non pandas.DataFrame type argument for the parameter 'data',
- giving a non list of strings for the parameter 'features',
- giving a non list of strings with at least one string not being one of the serie (column) within the dataframe 'data',
- a dataframe composed of categorical features only (string values
for examples like the features 'City' in the dataset).

Yes No

Basic tests

Performs the following basic tests with the given dataset
with at least 3 numerical features:
- run the method histogram with one, two and three valid features
- run the method density with one, two and three valid features
- run the method pair_plot with one, two and three valid features
- run the method box_plot with one, two and three valid features

Yes No

Exercise 07 - Komparator

The goal the exercise is to introduce plotting methods among the different libraries Pandas, Matplotlib, Seaborn or Scipy.

Error management

No specific output or behavior is expected if a parameters are wrongs or
if non expected parameters are given, nethertheless a basic management
is expected and those cases should be handled properly:
- giving a non pandas.DataFrame argument during the instance
of the object of class Komparator
- giving a string parameter which does not correspond to a serie of the dataframe (i.e. a random string) as 'categorical_var' parameter for the different methods
- giving a random string parameter as 'numerical_var' parameter for the different methods
- giving an existing numerical feature for 'categorical_var' for the different methods
- giving an existing categorical feature for 'numerical_var' for the different methods

☑ Yes ☒ No

Basic tests

Verify you get the expected plots for each methods:
- giving 'Age' and 'Medal' for the method compare_box_plots, you
should observe 3 boxes: Bronze. Silver and Gold
- giving 'Height' and 'Medal' for the method compare_histograms, you
should observe 3 curves on the histogram
- giving 'Weight' and 'Medal' for the method density, you should
observe 3 curves of density plot

☑ 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