Remember that the quality of the defenses, hence the quality of the of the school on the labor market depends on you. The remote defences during the Covid crisis allows more flexibility so you can progress into your curriculum, but also brings more risks of cheat, injustice, laziness, that will harm everyone's skills development. We do count on your maturity and wisdom during these remote defenses for the benefits of the entire community.

SCALE FOR PROJECT PISCINE PYTHON DJANGO (/PROJECTS/42CURSUS-PISCINE-PYTHON-DJANGO) / DAY 02 (/PROJECTS/42CURSUS-PISCINE-PYTHON-DJANGO-DAY-02)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.msk.21-school.ru:vogsphere/intra-uuid-11aede4d-c464-46
```

Introduction

For the smooth running of this evaluation, please respect the following rules:

- Remain polite, kind, respectful and constructive whatever happens during this conversation. It's a matter of confidence between you and the 42 community.

- Highlight the potential problems you 've had with the work you're presented to the person or the group you're grading, and take the time to talk about and discuss those issues.

- Accept the fact that the exam subject or required functions might lead to different interpretations. Listen to your discussion partner's perspective with an open mind (are they right or wrong ?) and grade them as fairly as possible.
42's teaching methods can make sense only if peer-evaluation is taken seriously.

Guidelines

- You must only evaluate what you will find in the student's or group's GiT repository.

- Take the time to check that the GiT repository matches the student or group and the project.

- Double check that no malicious alias was used to mislead you and make you grade something different from the official repository content.

- If a script supposed to help evaluate the exam is supplied by either side, the other side will have to strictly check it to avoid nasty surprises.

- If the evaluating student has not yet taken this project, they will have to

read the exam subject in its entirety before starting the evaluation.

- Use the flags available on this grading system to signal an empty or non-funcional project, a norm flaw, cheating, etc. In that case, evaluation stops and final grade is 0 (or -42 if it's a cheating problem). However, if it's not a cheating problem, you are invited to keep talking about the work that has been done (or not done, as a matter of fact) in order to identify the issues that lead to this stalemate and avoid it next time.

- You must stop grading when one exercise is not correct, even if the other ones are.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/22314/en.subject.pdf)

d02.tar.gz (/uploads/document/document/3388/d02.tar.gz)

Foreword

This section is dedicated to the evaluation start and the checking of prerequisites. It's not graded, but if something's wrong or a condition is not met, here or anytime during the evaluation, the grade is 0 and a flag can be ticked if necessary.

Observing the instructions

- The repo contains the evaluated student's or group's work.
- The evaluated student or group can explain their work anytime during the evaluation.
- General and specific instructions of the day will be observed during the whole evaluation.

Yes No

Python-Django training D00

- If you find an error in an exercise, the whole exercise is incorrect. No half-measure, no "slight error". The work is either perfect, either not. - The evaluated student must have provided tests for each Python file (block "if __name__ == '__main__':") unless specified otherwise in the subject. - Display of the pages must be managed: no strange behaviour, no special character (no accent) the display would not manage, etc. We want clean, or beyond.

Exercise 00 Conquering Silicon Valley!

- The render.py program properly manages the number of arguments (an error message appears if the program doesn't receive any argument or more than one).

- The program manages the occurencies where the file name in parameter doesn't have the .template extension (an error message appears).

- The program does write the result in a file which name is identical to the .template's one set in parameter. The only difference is an .html extension has replaced the .template one.

- The program converts the rendered template in an HTML file, the patterns are replaced by the values of the settings.py file and the informations required by the subject are presented in the produced HTML file.

- Modify the values included in the settings.py file, make sure the changes are taken into account when launching the program again.

- Required informations (Name, surname, page title, age, profession) must appear in the produced resume.

- The subject example is reproducible.

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

Yes No

Exercise 01 Innovating start-up looking for intern. 10 years exp. required.

- Your Intern class is present and can be instantiated with or without name set in parameter.

- The use of print() to display an instance allows to display the intance's "Name" attribute (the "name" set in parameter when instantiating or the default value).

- A Coffee class is present in the scope of the Intern class.The __str__() method is defined in the Coffee class as to returning "This is the worst coffee you ever tasted.".

- The work() method of the Intern class is present and raises an exception with the "I'm an intern, I can't do that..." text. The used type is the basic exception (Exception)

- The make_coffee() method is implemented in the Intern class and returns an instance of the Coffee class implemented in the scope of the Intern class (return self.Coffee()).

- Using print() on the return of the make_coffee() method (hence, an instance of the Intern.Coffee class) displays the return of the __str__() method of the Intern.Coffee class intance, that is "This is the worst coffee you ever tasted."

- The exception raised by calling the work() method is caught by a bloc try/except.

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

Yes No

ex02: 5 classes 1 cup.

- Each instance of the HotBeverage class or one of its derived classes is displayed the proper way:

name :
price :
description :

- The displayed informations for each instance of each class match the informations required by the subject.

- The content of each daughter class must be explicitly coded only if it's different from the mother class.
So if a price, name, or class class description is excplicitly rewritten in the daughter class whereas the included informations is identical to the one inherited from the mother class, the exercice is not considered valid.

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

Yes No

ex03: Glorious coffee machine!

- An EmptyCup class inheriting the HotBeverage is present in the scope of the CoffeeMachine class and the informations (description, price and name) match the informations provided with the subject.

- The CoffeeMachine class can be instantiated and the instance has a serve() method that takes in parameter a HotBeverage class or derived and a 50% chance of returning the instance of the class set as parameter, or 50% chance of returning an EmptyCUp instance. Make sure the return is random and required probabilities are met.

- The CoffeeMachine class has a builder.

- A BrokenMachineException class inheriting the Exception class also is implemented in the scope of the CoffeeMachine class. The message "This coffee machine has to be repaired." is defined in the builder.

- After calling the serve() method 10 times, a new call raises the CoffeeMachine.BrokenMachineException exception. Using print() on this expression allows the display of the message: "This coffee machine has to be repaired.".

- After calling the repair() method, the serve() method can be called 10 times again befor the machine "breaks down" again and raises the exception for each new call.

- Each exception raising is properly managed (= caught with a bloc try/catch) in your tests.

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

Yes No

Ex04: A basic class ft. RMS

- The test file in the appendix is running properly.

- The required structure in the subject is properly replicated and displayed in the tests.

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

Yes No

ex05: Create your own elements.

- The required classes are present and inherit all the Elem in the previous exercise. They're all functional.

- print(Html([Head(), Body()])) display:

- The structure required in the subject est properly replicated and displayed WITHOUT USING THE Elem CLASS.

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

Yes No

ex06: Validation.

- The Page class is present.

- print(Page(Html([Head(), Body()]))) displays:

- Page(Html([Head(), Body()])).is_valid() returns False.

- Page(Html([Head(Title(Text('title'))), Body()])).is_valid() returns True.

- Page(Html([Head(Title(Text('title'))), Body(Li())])).is_valid() returns False.

- Page(Html([Head(Title(Text('title'))), Body(Ol(Li(Text('foo')))))])).is_valid() returns True.

- Run as many tests as necessary. You MUST challenge the work turned-in by the student and test it as exhaustively as possible.

- Page(Html([Head(Title(Text('title'))), Body()])).write_to_file("test.html") create a test.html file containing:

```

```

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

Yes No

Ratings

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

✔ Ok

Empty work | No author file | Invalid compilation | Norme | Cheat | Crash

Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation