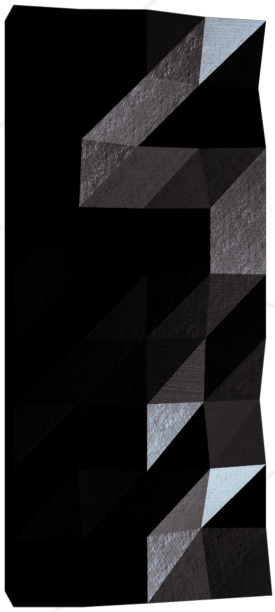


D00 - Django-Python Pool

Web Basics

Vincent Montécot vmonteco@student.42.fr
Damien Cojan dcojan@student.42.fr
Stéphane Ballet sballet@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: This first day will allow you to approach web development bases. On the menu : HTTP, HTML, CSS and integration of existing scripts in JavaScript your pages.

Contents

Chapter I

Preamble

Here is what Wikipedia says the *Balaenoptera musculus:*

The blue whale (*Balaenoptera musculus)* also known as blue whale is a species of cetaceans of the family *Balaenopteridae.* Can exceed 30 meters in length and 170 tons, is the largest animal living in our time and in the present state of our knowledge, the largest that ever lived on Earth.

Long and thin, the body of the blue whale can take various shades of bluish-gray on the back and a little more clear below. There are at least three subspecies dis- tinct: *B. m. musculus* in the North Atlantic and the North Pacific that, *B. m. intermedia* Antarctic Ocean *B. m. brevicauda* discovered in the Indian Ocean and in the south of the Paci fi c Ocean. *B. m. indica* discovered in the Indian Ocean, may be another subspecies. Like other whales, blue whales feed mainly a small crustacean, krill, but also small fish and sometimes squid.

Blue whales were abundant in nearly all oceans until the early twentieth century. For nearly forty years, they were hunted by whalers who brought the species to the brink of extinction before it is protected by the international community in 1966. A 2002 report estimated there were between 5 000 and 12,000 blue whales worldwide, located in at least five groups. More recent studies on the subspecies *B. m. brevicauda* suggest that this may be an underestimate. Before the industrial whaling, the largest population was in the Atlantic, which had approximately 240,000 (in '202 000 and 311

000). The species is considered threatened.

No whales were harmed during the writing of it.

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

Chapter III

Exercise 00

	Exercise: 00 00 Exercise: First
shell script	
Render Folder: *ex* 00 / Files to make: myawesomescript.sh	
Permitted functions: curl, grep, cut	
Remarks: n / A	

Yes Twitter has no secrets for you, you are very likely bit.ly :
a shortener handy URLs.

The purpose of this exercise is to write and make a shell script that ffi che the actual address of an address bit.ly (understand "the address to which the link returns bit.ly ").

As explicitly written in the cartridge of this exercise, you are not entitled to UTI Liser the following three shell commands to complete this exercise: curl, grep and
cut. Your best starting point is to read the manual command curl. To do this, type man curl in your device.

An example of the expected behavior of your shell script:

```
$> $ ./Myawesomescript.sh bit.ly/1O72s3U http://42.fr/>
```

The above example clearly shows that your script should be executable. interprets the user's / bin / sh.

Meet your script to a folder ex00 to the root of your deposit.

Chapter IV

Exercise 01

	Exercise 01 Exercise 01: Your
CV HTML	
Render Folder: *ex* 01 / files to make: cv.html	
Permitted functions: n / A	
Remarks: n / A	

You must create a CV HTML / CSS and meet the following requirements:

- You must respect the semantics of your HTML tags, and the separation of presentation and content.
- You must produce a file in HTML coherent structure with a minimum imposed content: name, skills and career.
- You need a ffi high to a title with the tag title and a title with the tag h1.
- You must use at least one table with tags table, th, tr and td.
- You must use at least one list with the tag ul and a list with the tag ol. The elements must use the tag li.
- The edges of the tables should be visible (solid). The edges of the tables should be merged (collapse).
- The lowermost cell to the right of each table must have # 424242 as border color.
- You will need to use a syntactic solution diff erent for each of the previous two rules: first use the tag style in the head your page. For the second, using an attribute style in a tag that you think suitable.

No special requirement for the veracity of information. You can make a wacky CV if you feel like it.

Chapter V

Exercise 02

	Exercise: 02
Exercise 02: Sending form of emails	
Render Folder: *ex* 02 / Files to make: form.html	
Permitted functions: n / A	
Remarks: n / A	

Make a form HTML which represents the usual information contact IN ANY. This form is to provide all of the following fields:

- firstname: a text field.
- name: a text field too.
- Age : you must use the specified numeric field as the HTML5.
- Phone: you must use the field as specified in that HTML5.
- E-mail : you must use the specific field email at HTML5.
- Student at 42? : you must use the checkbox field.
- Gender: you must use the radio buttons with values Male, Female and Other.
- A form submit button. The attribute onclick Your button must be: 'DisplayFormContents ().

the tarball d00.tar.gz annexed to this topic contains a subfolder EX02 / which itself contains a file Javascript popup.js written by the son of your boss internship in your company. As it would be unacceptable if you do spend the son of pa- tron for programming incompetant, you can not modify its fi le that should be used as such.

A careful reading and provided a superficial understanding of JavaScript is required to succeed this year.

You must properly integrate this fi le Javascript to your page HTML. If your code HTML is correct, pressing the Form button will bring up an ultramodern popup containing the fields and values of your form. In all other cases, your **code HTML is wrong.**

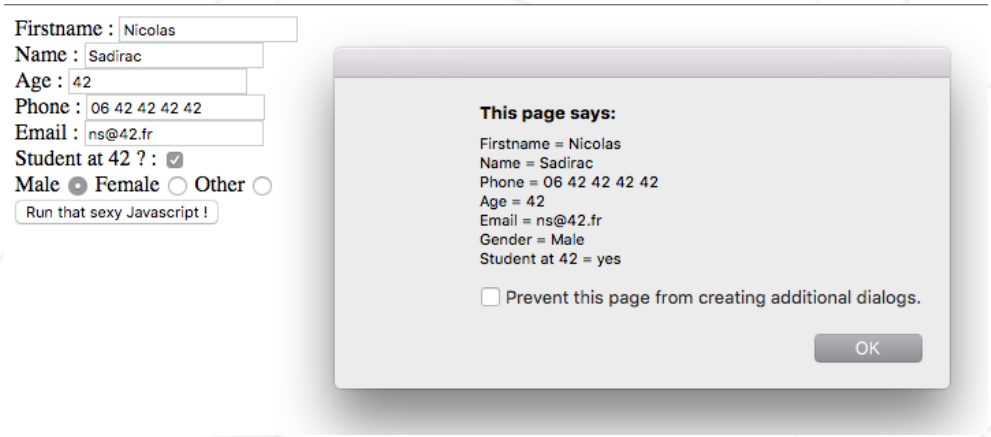


figure V.1 - noncontractual illustration of the expected result.

Chapter VI

Exercise 03

	Exercise: 03
Exercise 03: Reproduction of a web page	
Render Folder: *ex* 03 / Files to make: copy.html	
Permitted functions: n / A	
Remarks: n / A	

A competing company has posted a prettier webpage yours. With an industrial espionage worthy of Hollywood **cinema, your boss provides a screen shot of the page and the fi le css that goes with. These two fi les at your disposition in the Annexes to this in the archive d00.tar.gz and subfolder EX03 /.**

You must reproduce this as faithfully as possible page!

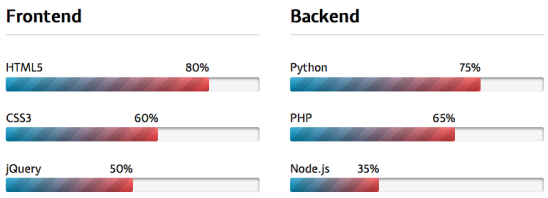


figure VI.1 - The screenshot of the page to duplicate. Scale of noncontractual image.

You will again separate the content and form, respect the semantics of tags used and maintain a logical structure in your document.

You must use the fi le css provided without the modifying. A "fresh" version of this css defense will be used to verify that you have complied with this order.

Chapter VII

Exercise 04

	Exercise: 04
Exercise 04: Integration of JS snippets.	
Render Folder: *ex* 04 / Files to make: snippets.html	
Permitted functions: n / A	
Remarks: n / A	

the tarball d00.tar.gz annexed to this topic contains a subfolder ex04 / which itself contains four fi les: file1.js, file2.js, file3.js and file4.js.

You must create and make a file snippets.html which must import all four scripts so that the pop-up was ffi che **correctly** (so no weird characters).

You can than import scripts in question, you can not modify or add Javascript in your HTML.

Chapter VIII

Exercise 05

	Exercise 05 Exercise 05:
W3C validation.	
Render Folder: *ex* 05 / Files to make: Your index.html corrected.	
Authorized Functions: Notes: n	
/ A	

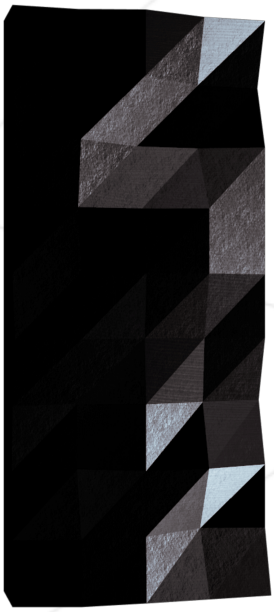
The code is good, the beautiful code is better, and to make good code, it is best to follow a good standard.

The W3C standard is a must in the field, and it is imperative to respect it when you write or you drive HTML.

You will find in the tarball d00.tar.gz annexed to this subfolder
ex05 / which contains the source of a complete webpage. Unfortunately, this page has been written by a developer much worse than you!

Correct the code HTML of the file index. html so that it passes the W3C validation ! This means, therefore, no errors and no *warning*.

You must *correct* the fi le and not truncate. This means that the contents of the file to correct *must* be present in your rendered in full.



D01 - Django-Python Training

Python bases 1

Damien Cojan dcojan@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: Today we discover together the first part of the bases syntactic and semantic Python.

Contents

Chapter I

Preamble

The Zen of Python, by Tim Peters Beautiful is
better than ugly. Explicit is better than implicit.
Simple is better than complex. Complex is better
than complicated. Flat is better than nested.
Sparse is better than dense. Readability counts.

Special cases are not special enough to break the rules. ALTHOUGH
practicality beats purity. Errors shoulds never pass silently. UNLESS
Explicitly silenced.

In the face of ambiguity, refuse the temptation to guess. There shoulds be one- and
preferably only one -obvious way to do it. ALTHOUGH That Way May not Be Obvious at fi
rst UNLESS you're Dutch. Now is better than never.

ALTHOUGH never Often is better than * right * now. If the implementation is hard to
explain, it's a bad idea. If the implementation is easy to explain, it May be a good
idea. Namespaces are one honking great idea - let's do more of Those!

import this

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- No code in the global scope. Make functions!
- Each fi le made must be completed by a function call in the same condition:

```
if __name__ == '__main__':
        your_function (whatever, parameter, is, required)
```

- It is permissible to place a error handling in the same condition.
- No authorized import, except for those explicitly mentioned in the 'Authorized Functions' of the cartridge each year ..
- The objections raised by the open function is not to manage.
- The interpreter is to use python3.

chapter IV

00 Exercise: my first variables

	Exercise: 00
00 Exercise: my first variables	
Render Folder: *ex* 00 / Files to	
make: var.py	
Permitted functions: n / A	
Remarks: n / A	

Create a named script var.py in which you must define a function my_var.
In it, declare new type variables and diff erent print on stan- dard output. You must reproduce exactly the following output:

```
$> Python3 var.py
42 is of type <class 'int'> 42 is of type <class 'str'> forty-two is of
type <class 'str'>

42.0 is of type <class 'float'> True is of type <class 'bool'>
[42] is of type <class 'list'> {42: 42} is of type <class 'dict'>
(42) is of type <class 'tuple'> set () is of type <class 'set'> $>
```

It is understood **not allowed** explicitly write types of your variables in your code prints. Do not forget to call your function at the end of your script as explained in the instructions:

```
if __name__ == '__main__':
        my_var ()
```

chapter V

01 Activity: Numbers

	Activity: 01 year 01:
Numbers	
Render Folder: *ex* 01 / files to make: numbers.py	
Permitted functions: n / A	
Remarks: n / A	

For this exercise, you are free to define as many functions as you want and name them as you like.

the tarball d01.tar.gz annexed to this topic contains a subfolder ex01 / wherein is a file numbers.txt containing 1 numbers 100 separated by a comma.

Design a script Python appointed numbers.py whose role is to open the fi le numbers.txt, read the numbers it contains, and the a ffi expensive to standard output, one per line without commas.

chapter VI

Exercise 02: My first dictionary

	Exercise: 02
Exercise 02: My first dictionary	
Render Folder: *ex* 02 / Files to make: var_to_dict.py	
Permitted functions: N / A	
Remarks: n / A	

You are free to define as many new features as you want and name them as you like. This set will no longer mentioned, except in cases of explicit contradiction.

Create a named script var_to_dict.py in which you need to copy the list of pairs d such in the next one or another of your duties:

```
d = [
    ( 'Hendrix', '1942'), ( 'Allman'
                                ,    1946),
    ( 'King'                    ,    '1925'),
    ( 'Clapton', '1945'), ( 'Johnson', '1911'), (
    'Berry'
                                ,    '1926'),
    ( 'Vaughan', '1954'), ( 'Cooder'
                                ,    '1947'),
    ('Page'                     ,    '1944'),
    (Richards', '1943'), ( 'Hammett', '1962'),
    ( 'Cobain'
                                ,    '1967'),
    ( 'Garcia'                  ,    '1942'),
    (Beck '                     ,    '1944'),
    ( 'Santana', '1947'), ( 'Ramone'
                                ,    1948),
    ( 'White'                   ,    '1975'),
    ( 'Frusciante', '1970'), ( 'Thompson',
    '1949'), ( 'Burton'
                                , '1939')
]
```

Your script must transform this variable in a dictionary with the key date, and the name of the musician for value. It must then a ffi expensive this dictionary on standard output as the precise formatting:

```
1970: Frusciante 1954:
Vaughan 1948: Ramone
1944 Page Beck 1911
Johnson

. . .
```

the final order will not necessarily identical to the example, and this is normal behavior. Do you know why ?

chapter VII

Exercise 03: Search by keywords

	Exercise 03 Exercise 03:
Search by keywords	
Render Folder: *ex* 03 / Files to make: capital_city.py	
Permitted functions: import sys	
Remarks: n / A	

You have the following dictionaries to copy such in one or the other functions of your script:

```
states = {
    "Oregon"                : "GOLD",
    "Alabama", "G", "New Jersey":
  "NJ", "Colorado": "CO"}



capital_cities = {
    "OR": "Salem", "G",
  "Montgomery", "NJ": "Trenton",
  "CO": "Denver"}
```

Write a program that takes as argument a state (ie Oregon) and a ffi che on output standand its capital (eg Salem). If the argument does not work, your script should a ffi expensive: Unknown state. If there is not, or if too many arguments, your script should do and leave nothing.

```
$> Python3 capital_city.py Oregon Salem

$> Python3 capital_city.py Ile-De-France Unknown state

$> Python3 capital_city.py
$> $ Python3 capital_city.py Oregon Alabama> python3 capital_city.py Oregon Alabama
Ile-de-France $>
```

chapter VIII

Exercise 04: Search value

	Exercise 04 Exercise 04: Search
value	
Render Folder: *ex* 04 / Files to make: state.py	
Permitted functions: import sys	
Remarks: n / A	

You have again the same two dictionaries than last year. You must copy them again as such in one or the other functions of your script.

Create a program that takes a capital in argument and ffi che this time Ci the corresponding state. The rest of the behavior of your program should be identical to those of the previous year.

```
$> Python3 state.py Salem Oregon

$> Python3 state.py Paris Unknown capital
city $> $ python3 state.py>
```

chapter IX

Exercise 05: Search by keywords or by value

	Exercise: 05
Exercise 05: Search by keywords or by value	
Render Folder: *ex* 05 / Files to make: all_in.py	
Permitted functions: import sys	
Remarks: n / A	

By always starting from the same two dictionaries, you should always copy such in one or the other functions of your script, write a program with the following behaviors:

- The program should take as argument a string containing all expressions to search you want, separated by commas.
- Each expression of this string, the program should detect if it is a capital city, a state, or neither.

- The program should not be case sensitive, or white spaces too.
- If there is no parameter or too many parameters, the program has nothing ffi che.

- When there are two commas in a ffi Lées in the string, the program has nothing ffi che.
- The program needs a ffi expensive results separated by a newline. and using the following specific format:

```
$> Python3 all_in.py "New Jersey, Trenton, toto,                    ,          Salem "
Trenton is the capital of New Jersey Trenton is the capital of New
Jersey foo Neither is a capital city nor state has Salem is the capital of
Oregon $>
```

chapter X

Exercise 06: Sort a dictionary

	Exercise 06 Exercise 06: Sort a
dictionary	
Render Folder: *ex* 06 / Files to make: my_sort.py	
Permitted functions: N / A	
Remarks: n / A	

Integrate the dictionary with any of your duties as such:

```
d = { 'Hendrix': '1942'

        'Allman'                '1946',
        'King'                  '1925'
        'Clapton': '1945', 'Johnson' '1911' 'Berry'

                                '1926'
        'Vaughan': '1954', 'Cooder'
                                '1947',
        'Page'                  '1944'
        Richards', '1943', 'Hammett': '1962',
        'Cobain'
                                '1967',
        'Garcia'                '1942'
        Beck '                  '1944'
        'Santana', '1947', 'Ramone'
                                '1948'
        'White'                 '1975'
        'Frusciante' '1970,' 'Thompson' '1949,'
        'Burton'
                                '1939'
}
```

Write a program that ffi che names of musicians sorted by ascending order of year and in alphabetical order of names when the dates are identical, one per line, without a ffi expensive dates.

chapter XI

Exercise 07: Periodic Table

	Exercise: 07
Exercise 07: Periodic Table	
Render Folder: *ex* 07 / Files to make: periodic_table.py	
Permitted functions: import sys	
Remarks: n / A	

the tarball d01.tar.gz annexed to this topic contains a subfolder EX07 / wherein is a file periodic_table.txt which describes the periodic table of ele- ments in a convenient format for IT professionals.

Create a program that uses this fi le to write one page HTML representing the periodic table of elements formatted properly.

- Each element must be a 'case' of a table HTML.
- The name of an item must be in a title tag level 4.
- The attributes of an element should be in list form. Must be listed at least the atomic number, symbol, and atomic mass.
- You must comply with minimum layout of Mendeleev table as we can find it on google, namely that there must be empty boxes where there must have, as well as linebreaks here where it takes. Your program should create the fi le result periodic_table.html. This fi le HTML

must of course be immediately readable by any browser and must be valid W3C.

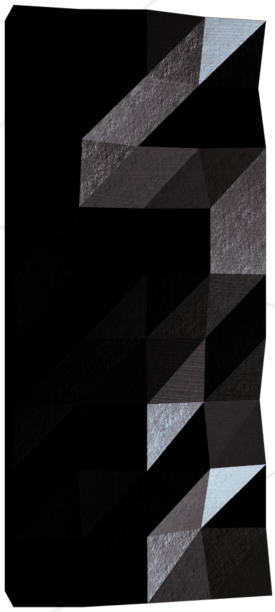
You are free to design your program as you wish. Feel free to split your code into separate and possibly reusable functionality. You

can customize the tags with a CSS style "inline" to make your made more attractive (if only that the contour of the table spaces), or even generate a file
periodic_table.css if you prefer.

Here is an excerpt of sample output to give you an idea:

```
[...]
   <Table>
      <Tr>
         <Td style = "border: 1px solid black; padding: 10px">
            <H4> Hydrogen </ h4>
                  <Ul> <li> No 42 </ li>

                        <Li> H </ li> <li> 1.00794 </ li>
                  <li> one electron </ li> <ul> </ td> [...]
```



D02 - Django-Python Training

Python Basics 2

Vincent Montécot vmonteco@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: Today you are going to conquer Silicon Valley through your new skills in OOP with Python!

Contents

Chapter I

Preamble

These are the words of the "Free Software Song": Join us now and share the software; You'll be free, hackers, you'll be free. Join us now and share the software; You'll be free, hackers, you'll be free. Hoarders can get piles of money, That is true, hackers, That Is true. Purpose They Can not help Their neighbors; That's not good, hackers, that's not good. When we-have enough free software At our call, hackers, at our call, We'll kick out Those dirty licenses Ever more, hackers, ever more. Join us now and share the software; You'll be free, hackers, you'll be free. Join us now and share the software; You'll be free, hackers, you'll be free.

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- No code in the global scope. Make functions!
- Unless otherwise indicated, all written in Python fi les will end with a block

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

- Any exception catchée not invalidate the work, even in a case of error you are asked to test.
- No authorized import, except for those explicitly mentioned in the 'Authorized Functions' of the cartridge each year ..

Chapter IV

Exercise 00

	Exercise: 00
Exercise 00: Conquering the Silicon Valley!	
Render Folder: *ex* 00 / Files to make: render.py, myCV.template, settings.py	
Permitted functions: import sys, os, re	
Remarks: n / A	

You have fi nished your super developer training and a new life full of S'o ff ers prospects to you. Arriving in Silicon Valley, you have an idea in mind: develop your idea of revolutionary CV generator using a pioneering technology, and become the new Bill Gates of the job search.

It only remains to develop the technology. Make a program render.py which will take a file with an extension. template

parameter. This program will read the contents of the file, change some patterns with defined values in a file settings.py (The presence of a block if __name__ == '__main__': is not required for this fi le) and write the result to a file with the extension. html.

The following example *will* be exactly reproduced with your program.

```
$> Cat settings.py: name =
"duoquadragintian" $> cat file.template <p>
"- Who are you?

- A {name}! "</ P>
$> $ Python3 render.py file.template> cat file.html <p> "-
Who are you?

- A duoquadragintian! "</ P>
```

The mistakes, including poor fi le extension, a file does not exist or a bad number of arguments to be managed.

You must make a file myCV.template which, when converted to HTML fi le, should at least contain the complete structure of a page (doctype, head and body), the page title, the name and surname of the owner of CV, age, and profession. Of course, this information should not appear directly in the fi le

. template.

Chapter V

Exercise 01

	Exercise: 01
01 Exercise: Innovative Startup seeks intern. 10 years of experience required. Render Folder: *ex* 01 / files	
to make: intern.py	
Authorized Functions: Notes: n	
/ A	

You can not go it alone on such an adventure. You decide to recruit someone to do ter coffee assist you, preferably a trainee (it's ~~cheaper~~).

Realize the class Intern containing the following features:

A builder taking a string parameter and assigning its
an attribute value Name. A default " My name? I'm nobody, an intern, I have no name. " will be implemented.

A method __ str __ () which will return the attribute name of the proceedings.

A class Coffee with a simple method __ str __ () which will return the string
character " This is the worst coffee you ever tasted. ".

A method work () which will lift just one exception (use type (Exception)
Basic) with the text " I'm just an intern, I can not do that ... ".

A method make_coffee () which will return an instance of the class Coffee than
you implemented in the classroom Intern.

In your tests, you need to instantiate the class twice Intern once nameless, and once with the name "Mark".

A ffi in the name of each instance. Ask Mark make you a coffee and AF-fi in the result. Ask the other student to work. You **must** handle the exception in your test.

Chapter VI

Exercise 02

	Exercise 02 Exercise 02: 5
classes 1 cup.	
Render Folder: *ex* 02 / Files to make: beverages.py	
Authorized Functions: Notes: n	
/ A	

The coffee is good but in the long run it's a bit boring to not have more choices. Make a class HotBeverage with the following features:

An attribute price a value of 0.30.

An attribute name with the value "Hot beverage".

A method description() returning a description of the proceedings. The value
the description will be " Just some hot water in a cup. ".

A method __ str __ () returning a description of the proceedings in this form:

```
name: <name attribute>
price: <price attribute limited to two decimal point> Description: <instance's description>
```

eg a ffi chage an instance of HotBeverage give:

```
name: hot beverage price: 0.30

Description: Just some hot water in a cup.
```

Realize then derived classes HotBeverage following:

Coffee:

name: " coffee "

price: 0.40

description: " A coffee, to stay awake. "

Tea:

name: " tea "

price: 0.30

description: " Just some hot water in a cup "Chocolate.:

name: " chocolate "

price: 0.50

description: " Chocolate, sweet chocolate ... "Cappuccino:

name: " cappuccino"

price: 0.45

description: "Un po 'di Italia nella sua Tazza"

You must redefine THAT what is necessary. (Cf. DRY)

Instantiate in your tests from each class: HotBeverage, Coffee, Tea, Chocolate and Cappuccino and a ffi in.

Chapter VII

Exercise 03

	Exercise: 03
Exercise 03: Glorious co ff ee machine!	
Render Folder: *ex* 03 / Files to make: machine.py, beverages.py	
Permitted functions: import random	
Remarks: n / A	

That's it, your company is on! You now have a local acquired through your first fundraiser, a trainee to make coffee and a level of green plant 10 to the entrance of the building to keep it all.

However it must be admitted: coffee produced by your intern is rubbish and a half minimum wage per month is a bit expensive for the sock juice. This is the time of in- vestir in new equipment necessary for your professional success!

Realize the class CoffeeMachine containing:

- A builder.
- A class EmptyCup inheriting HotBeverage, with the name " empty cup "
  as prices 0.90 and the description " An empty cup ?! Gimme my money back! ".
  Copy the fi le beverages.py last year in the record of this exercise to use the classes it contains.
- A class BrokenMachineException inheriting Exception with the text " This coffee machines Has to be repaired. ". This text must be defined in the constructive tor of the exception.
- A method repair () which repairs the machine for it to be used to nou- calf hot drinks.
- A method serve () which will have the following characteristics:

  Settings : A single parameter (other than self) which will be a derived class
  of HotBeverage.

  return: Once two (randomly), the method returns an instance of

last classroom setting, and half the time an instance of EmptyCup.

obsolescence: The machine is not of the best quality and therefore falls failure after 10 drinks served.

In case of failure : the method call serve () should cause the removal of a exception type CoffeeMachine.BrokenMachineException up to-that the method repair () is called.

repair: After a call to the repair (), the method serve () can again operate without exception thrown during a cycle of 10 drinks before falling down.

In your tests, instantiate the class CoffeeMachine. Ask various drinks ing ve- fi le beverages.py and a ffi in the drink you used the machine until it fails it (you *must* handle the exception then lifted). Fix the machine and start to do the machine breaks down again (except manage again).

Chapter VIII

Exercise 04

	Exercise: 04
Exercise 04: A base class ft. RMS.	
Render Folder: *ex* 04 / Files to make: elem.py	
Authorized Functions: Notes: n	
/ A	

Now is the time to improve your presence on the web. You would like to use your new skills in Python for modeling e ff ectively your HTML content, but you want to receive the advice of a superior being to find out how. You decide to o ff er your intern sacrifice that the gods of programming.

Now that you have a machine to make coffee, you do not really need it ... You therefore immolate.

Saint IGNUcius you will appear to you a revelation:

> *"HTML elements share more or less the same structure (tag, content, attributes). It would be wise to conduct a class capable of Ras seem all these behaviors and characteristics common to then use the power of inheritance in Python to derive easily and sim- plement this class without having to rewrite everything. "*

Then St. IGNUcius see the Mac on which you work. Taking fear, he fled without giving you more details, leaving behind only a fi le testing. Without hesitation, you realize the class Elem with the following characteristics:

- A builder may take as parameter the name of the element, its attributes HTML, its content and the element type (single or double tags).
- A method __ str __ () returning the code HTML of the element.
- A method add_content () to add items to the end of the content. If you are doing your job well, you will be able to represent any element HTML and content with your class Elem. Home stretch :

- the fi le tests.py tarball provided in the annex of the subject **must** operate cor- rectly (no assertion error, the output of the test explicitly announcing his success). Obviously, we are not cruel enough to test the functionalities that are not explicitly requested in this exercise. Hahaha ... No, we are not, I assure you.

- You will also need to reproduce and expensive ffi the structure with your class Elem:

```
<Html>
    <Head>
        <Title>
            "Hello ground!" </ Title>
     </ head> <body>


        <H1>
            "Oh no, not again!" </ H1>

        <Img src = "http://i.imgur.com/pfp3T.jpg" /> </ body> </ html>
```

Chapter IX

Exercise 05

	Exercise: 05
Exercise 05: Make your own items!	
Render Folder: *ex* 05 / Files to make: elem.py, elements.py	
Authorized Functions: Notes: n	
/ A	

Congratulations! You are now able to generate any HTML element and its contents. However, this is a bit cumbersome to generate each item by stating where each attribute to each instantiation. This is an opportunity to **use inheritance to other little easier to use classes. Perform the following classes derived from your class Elem last** year:

- html, head, body
- title
- meta
- img
- table, th, tr, td
- ul, ol, li
- h1
- h2
- p
- div
- span
- hr
- br

The constructor of each class will be able to take content first argu- ment, as well:

```
print (Html ([Head () Body ()]))
```

a ffi chera well:

```
<Html>
    <Head> </ head>
  <body> </ body> </
  html>
```

Be smart and reuse functionality you encoded in the class Elem. You **must** use inheritance.

Demonstrate the operation of these classes for testing your desired number su ffi cient to cover all the features. After having addressed these classes, you will not need to specify the name or type of a tag, which is very convenient. Therefore you should never instantiate your class Elem, It is now **not allowed.**

To help you understand the advantage derived classes Elem compared to the direct use Elem, resume the document structure HTML the previous exer- cise. You must reproduce it using your new classes.

```
<Html>
    <Head>
        <Title>
            "Hello ground!" </ Title>
      </ head> <body>


        <H1>
            "Oh no, not again!" </ H1>

        <Img src = "http://i.imgur.com/pfp3T.jpg" /> </ body> </ html>
```

It's much more simple, right? :)

Chapter X

Exercise 06

	Exercise 06 Exercise 06:
Validation	
Render Folder: *ex* 06 / Files to make: Page.py, elem.py, elements.py	
Authorized Functions: Notes: n	
/ A	

Despite real progress in your work, you would like that everything is a little cleaner. A little framed, you are like that: you like the constraints and challenges. So why not impose a standard structure for your documents HTML? Commencez by copying the classes of the two previous years in the record of this exercise.

Create a class Page which the manufacturer must take as a parameter an instance of a class inheriting Elem. Your class Page must implement a method isValid ()
which must return true if all the following rules repectées and false if not :

- If during the course of the tree, a node is not type html, head, body, title, meta, img, table, th, tr, td, ul, ol, li, h1, h2, p, div, span, hr br or
  text, the shaft is invalid.
- html must contain exactly Head, then a Body.
- Head should only contain a single title and only this Title.
- Body and div only contain elements of following type: H1, H2, Div, table, ul, ol, Span, or Text.
- Title, H1, H2, Li, Th, Td must not contain a single text and only this
  Text.
- P should contain only Text.
- span should contain only text or some P.
- IU and ol] must contain at least one Li and only Li.

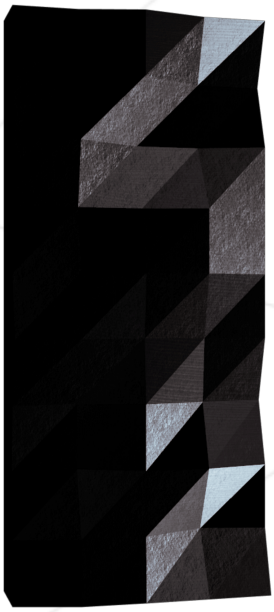
- **Tr must contain at least one Th or td and only Th or some Td. The Th and the td must be mutually** exclusive.
- **table: should contain only Tr and only Tr.**

Your class Page must also be able to:

- **A ffi expensive its code HTML when it print an instance. Warning: the code HTML**
  a ffi ket must be preceded by a doctype if and only if the type of the root element Html.

- **Write the code HTML in a file with a method write_to_file which takes as a parameter the name of the file.**
  Warning: the code HTML written in the fi le must be preceded by a doctype if and only if the type of the root element
  Html.

Demonstrate how your class Page by tests of your choice by many su ffi cient to cover all the features.



D03 - Django-Python Training

Python - libraries

Damien Cojan dcojan@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: Today we'll handle some practical bookstores Python.

Contents

Chapter I

Preamble

Geohashing From Wikipedia, the free encyclopedia

Geohashing is an outdoor recreational activity inspired by the webcomic xkcd, participants in qui-have to reach a random rental (Chosen by a computer algorithm), prove Their achievement by Taking a picture of a Global Positioning System (GPS) recei- ver Mobile Reviews another gold device and Then tell the story of Their trip online. Proof is not based electronic navigation est acceptable.

Whereas other outdoor recreational activities like geocaching Have a precise goal, geo- hashing is Mainly fueled by ict pointlessness, Which is deemed funny ict by players. The resulting and geohashing community and culture is extremely THUS tongue-in-cheek, suppor- ting any kind of humorous behavior During the practice of geohashing and resulting and in a parody of traditional outdoor activities. Navigating to a random spot need not be point-less. Some paper geohashers new mapping features They fi nd on the OpenStreetMap project.

Source Wikipedia

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.

- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.

- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...

- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- No code in the global scope. Make functions!
- Each fi le made must be completed by a function call in the same condition:

```
if __name__ == '__main__':
        your_function (whatever, parameter, is, required)
```

- It is permissible to place a error handling in the same condition.
- No authorized import, except for those explicitly mentioned in the 'Authorized Functions' of the cartridge each year ..
- The interpreter is to use python3.

Chapter IV

Exercise 00

	Exercise: 00 Exercise 00:
Antigravity	
Render Folder: *ex* 00 / Files to make: geohashing.py	
Permitted functions: sys and antigravity modulus	
Remarks: n / A	

This is a small exercise échau ff ly echoing the preamble of the day. Nothing complicated.

Make a small program named geohashing.py that takes as many parameters as needed to calculate a typical Geohash and therefore must obviously calculate this Geohash before a ffi expensive standard output.

On error, the program has a ffi expensive a relevant message you chose before exiting cleanly.

This diagram can, perhaps, help: Geohashing algorithm

Chapter V

Exercise 01

	Activity: 01 year 01:
Pip	
Render Folder: *ex* 01 / files to make: my_script.sh my_program.py	
Permitted functions: path.py Module	
Remarks: n / A	

path.py is a library that implements a Path object around the module os.path
Python, making it very intuitive.

In this exercise, you create a script bash that installs this library, and a program Python who is using it.

The shell script must meet this description:

- Its name must have the extension. sh because it is a script Shell.
- It is a ffi expensive version used pip.
- It must install the development version path.py since its repo GitHub,
  in a file that must be called local_lib, placed in the output folder. If the library has already been installed in this folder, the installation must then crush him.
- He must write the installation log path.py in a file having the extension
  . log
- If the library has been properly installed, it must ultimately carry the small pro- gram that you must also create.

The program Python to create is a composition of your choice, which must nevertheless comply with these restrictions:

- Its extension should be. py because it is a program Python.
- It must import the module path.py from where this library is installed, thanks to the previous script.
- It should create a folder and a file on within that folder, write something in this fi le and then finally read a ffi expensive content.
- It must respect the specific rules of the day c.

Chapter VI

Exercise 02

	Exercise 02 Exercise 02:
querying an API	
Render Folder: *ex* 02 / Files to make: request_wikipedia.py requirement.txt	
Permitted functions: modules requests, JSON, and dewiki sys	
Remarks: n / A	

Wikipedia is a great tool shared you know necessarily. It is dis- ponible on your favorite browser and even as mobile application. I propose now to create a tool that allows you to query the site now become essential, right from your device.

To do this, you must design a program named request_wikipedia.py
that takes a parameter string and e ff ectue a search through the API Wikipedia
before writing the result in a file. You can choose to query the API English or French.

- The program needs to write a result, even if the query is misspelled. Take the original site for example, if it finds you a result for a given request, then your program too.
- The result should be free of any formatting JSON or Wiki Markup before being written in the fi le.
- The name of the file must have the format nom_de_la_recherche.wiki and must not contain any space.
- If no parameters, wrong settings, invalid application of in- formation not found, server problem or any other problem: no expensive fi should be created and an appropriate error message to be a ffi ket on the console.
- Include in your rendering a file requirement.txt that will be used in defense to install the necessary libraries to your program in a virtualenv

or on the system.

The dewiki library is not perfect, we do not seek the cleanest possible result, it is not the purpose of this exercise.

Read the API documentation. Observe the structure of what you will be returned.

Here's an example of what we expect:

```
$> Request_wikipedia.py python3 "chocolatine" $> cat chocolatine.wiki




A chocolatine means:
*   a pastry with chocolate, also called chocolate croissant or chocolate couque;
*   a pastry has patissiere cream and chocolate, also called Swiss;
*   a kind of chocolate candy;
*   a book of Anna Rozen


Despite its ancient usage, the word is not in the dictionary between Webster in 2007 and in the Petit Larousse in 2011.


The use of the term "Chocolatine" is also found in Quebec, whose language has evolved from
    old french french differently from employees in Europe, but this usage proves nor denies any anticipation, dependent chance to use the first trader having introduced in
 Quebec. References




Category: Confectionery Category:
Chocolate
```

Chapter VII

Exercise 03

	Exercise 03 Exercise 03:
Parser html	
Render Folder: *ex* 03 / Files to make: roads_to_philosophy.py, requirement.txt	
Permitted functions: Requests and BeautifulSoup modules	
Remarks: n / A	

Legend says that if you go to any Wikipedia article, you click on the first link in the introduction to this article is neither italicized nor brackets, then you repeat this process loop, you aboutirez invariably on Article Philosophy.

Well aware that this is not a legend! (Make 'Oooooh!' With a bewildered air). Witness this article ... of Wikipedia.

However, as you believe what you see with your own eyes, you **must** create a program that puts this to the test by listing then COMP as all items between your query and the Wikipedia: the **roads to philosophy.**

This program must be named roads_to_philosophy.py and have the following behavior:

- The program must take a string parameter which is a word or set of words corresponding to a single Wikipedia search.
- The program must querying a URL of Wikipedia **English** identical to that of a standard search on a browser. In other words, it is not **not allowed** of UTI Liser API site.

- It must parse the page html through the library BeautifulSoup in order to:
  - Find the possible redirection, and take into account in **roads to philosophy.** Warning, it is not about url redirection.
  - Find the main title of this page and add it to **roads to philosophy.**
  - Find (if it exists) **the first link in the introductory paragraph** condui- health to another Wikipedia article. Rather than ignoring what is in italics and brackets, the program must **ignore** Carefully links that do not point to a new article, such as those leading to the help section Wikipedia.

- The program should start from the **2nd step** obtained starting from the link to the previous step to fall on one of the following cases:
  - The link leads to the page philosophy, in which case it should print on the standard output sections visited and the total consideration of these items in the format < number> roads from <application> to philosophy
  - The page contained no valid links The program has a ffi expensive: It leads to a dead end!.
  - The link leads to a previously visited page, which means that the program is ready to wrap ap- inde fi rect handling. In this case the message ffi ket must be: It leads to an infinite loop!

- At this stage, after a ffi ket on the standard output the necessary messages, the program should exit gracefully.

If at any time in the execution of the program, an error occurs, such as an error connecting to server, setting, request or other: the program should exit gracefully with an appropriate error message.

As in the previous year, vou will need to provide your program with an expensive fi requirement.txt in order to facilitate the installation of libraries.

The output of your program should look like this:

```
$> Python3 roads_to_philosophy.py "42 (number)" 42 (number) number Natural
Mathematics Ancient Greek Greek Language Modern Greek colloquialism Word




Linguistics Science Knowledge
Conscious Awareness
Consciousness Quality
(philosophy) Philosophy



17 roads from 42 (number) to philosophy! $> Python3
roads_to_philosophy.py Accuvio It's a dead end! $>
```

The Wikipedia community regularly puts its products up to date. It is possible that between the time of writing this and that to which you do this training, the example accuvio no longer a dead end.

Chapter VIII

Exercise 04

	Exercise 04 Exercise 04:
virtualenv	
Render Folder: *ex* 04 / Files to make: requirement.txt my_script.sh	
Permitted functions: all	
Remarks: n / A	

Tomorrow is the first day of training on the framework Django. You need to prepare the ground by configuring a little easy installation.

You have to create two elements:

- A file requirement.txt which must contain the latest stable versions of django and of psycopg2.
- A script must have this behavior:
  - Have the extension. sh
  - To create a virtualenv under python3 appointed django_venv.
  - Install the fi le requirement.txt you created in this Virtualenv.
  - Leaving the virtualenv must be **activated.**

Chapter IX

Exercise 05

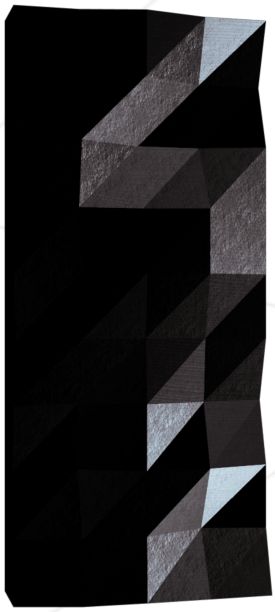
	Exercise 05 Exercise 05:
Hello World	
Render Folder: *ex* 05 / Files to make: all necessary file	
Permitted functions: all	
Remarks: n / A	

Simply install Django must be frustrating, and that's perfectly understandable.

So you will end this day on the beauty in bookstores by designing your first Bonjour Monde with Django.

In the last year, you must monitor and adapt the tutorial o ffi sky in order to make a web page, from the address http: // localhost: 8000 / helloworld, a ffi che simply text Bonjour Monde ! in your browser.

Your made should be a folder containing the project Django.



D04 - Django-Python Training

Things are getting serious.

Vincent Montécot vmonteco@student.42.fr
42 Sta ff pedago@staff.42.fr

Abstract: Python, Django!

Contents

Chapter I

Preamble

Here is the list of monsters that you can encounter while exploring a dungeon Earth Fangh:

- all kinds of undead.
- giant spiders.
- orcs and goblins.
- Troll in the underground.
- witches.
- cursed warriors.
- giant rats.
- a bottle of oil.
- toilet paper.
- two sponges.
- ravioli.

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.

- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.

- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...

- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- The roads on an application must be defined in a file urls.py
  found in the record of this application.
- Any form (derived class django.forms.Form) must be in the fi le forms.py of the application to which it is
  attached.
- Each page has ffi chée must be properly formatted (presence of a doctype, tags couples html, body, head) Correct
  management of special characters, not a ffi weird chage.
- The server used to this day is the default Django development server included with the utility manage.py.
- Only explicitly requested URLs must return an error page. Thus, if only / ex00 is requested / ex00foo should
  return a 404 error.
- The requested URL must function with and without slash purpose. So if / ex00
  is requested / ex00 and / ex00 / must both work.

Chapter IV

Exercise 00

	Exercise: 00
00 Exercise: First static page.	
Render Folder: *ex* 00 / Files to make: requirements.txt, files and folders of your project and your application	
Permitted functions: All the features of Django.	
Remarks: n / A	

You go into this exercise make your first static page with Django. Create virtualenv with python3, install Django create

a file requirements.txt
containing project dependencies (which is installed in the virtualenv with pip) at the root of the rendering.

Start a project d04.
Start an application ex00.

Make a page that will be available at the URL / ex00 your site. Gather in this page for information on all of the syntax markdown, and give the title to this page " Ex00: Markdown Cheatsheet. ".

The template used should be named index.html.

Chapter V

Exercise 01

	Exercise: 01
Exercise 01: A few more pages.	
Render Folder: *ex* 01 / files to make: requirements.txt, files and folders on your project and your application.	
Permitted functions: All the features of Django.	
Remarks: n / A	

Perform the following pages in a second application ex01:

- **Title:** "Ex01: Django web framework."

  URL: / ex01 / django.

  Description: Make this page a brief presentation of Django and his historical.

- **Title:** "Ex01: Process a ffi chage a static page."

  URL: / ex01 / Display

  Description: Describe this page in the process leading to a ffi a chage static web page from a simple template through a view of the request to the response.

- **Title:** "Ex01: Template engine."

  URL: / ex01 / templates

  Description: Describe this page in the engine operating tem- default flat Django and operation:

  - Blocks.
  - Curls for ... in.
  - Control structures yew.

- From a ffi variables chage past by context. The principle of *do not repeat yourself* an integral part

of the philosophy of Django. To observe this principle, realize the requested pages using a basic template named base.html.

the template base.html must include:

- A block happy in the body.
- A block style in the head.
- A block title in the head.

Realize also a template nav.html containing a navigation bar that lists links to each page of the exercise.

All pages of this exercise should be based on the template base.html. The tem- plate nav.html must be included in all pages.

Also create two fi les style1.css and style2.css. The first will define the text color as the *blue,* the other will define as the *red.*
You will need style1.css on every page except the "template engine" where you will use style2.css.

You should use each one style sheet and only time in all of your templates for this exercise.

Chapter VI

Exercise 02

	Exercise 02 Exercise 02: First
form.	
Render Folder: *ex* 02 / Files to make: requirements.txt, files and folder of your project and your application.	
Permitted functions: All the features of Django.	
Remarks: n / A	

Make a page accessible through the URL / EX02 in a new application EX02.

This page will contain two parts:

- A form with a text field and a button Submit.
- One part to contain the history of inputs. The rule concerning the

form:

- You must not NOT encode the form field hard, but use the class django.forms.Form that provides Django to create a form. You pass the ground as context the rendering function of the template.

Here are the rules for the history:

- The history entries will initially be empty.
- Each text by submitting the form you will need:
  - Register input and its time stamp at the end of a file of logs. If this fi le does not exist it is automatically created. This fi le must be created in the application folder hand.
  - Add the entry to the history of the page, with the time and date of submission.
- The path to the fi le logs (name of the file included) must be defined in the fi le settings.py of the project. Thus, each entry submitted will be present in the historical part of the page and in the fi le log, preceded by its time stamp.
- Data must also be persistent. If the development server must be restarted for some reason, the data should not be per- due.

Chapter VII

Exercise 03

	Exercise 03 Exercise 03: Fifty
Shades of bic.	
Render Folder: *ex* 03 / Files to make: requirements.txt, files and folders on your project and your application.	
Permitted functions: All the features of Django.	
Remarks: n / A	

Create a last application EX03.

A ffi to a page containing an array of 4 columns and 51 lines (one line to the column names).

This page will be accessible at the URL / EX03.

Each column of the table correspond to a color among *black, red, blue, green.*

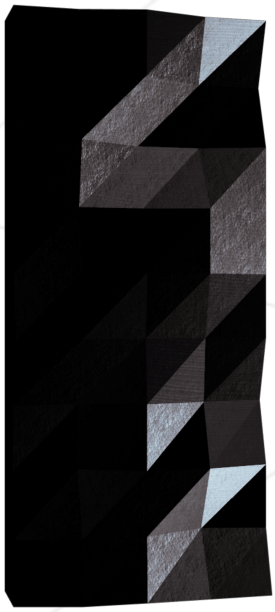
The cells of the table must have the following characteristics:

Height: 40 pixels.
Width : 80 pixels.
Background color : a shade of color that matches the column.

The table is a ffi expensive shades of each color to obtain a gradient of 50 lines.

You must observe the following:

- Your template must NOT contain color hard. The di ff erent shades must be generated in one view and all be di ff erent.
- 4 pairs of tags < td> </ td> are allowed in your template.
- 4 pairs of tags < th> </ th> are allowed in your template.
- Your table should be formatted like this:
  - A couple of tags < th> </ th> by box to the column names.
  - A couple of tags < tr> </ tr> color shade line.
  - A couple of tags < td> </ td> color swatch box.



D05 - Formation Python-Django

Django - ORM

Damien Cojan dcojan@student.42.fr
42 Staff pedago@staff.42.fr

Résumé: Aujourd'hui nous allons découvrir l'ORM de `Django`.

Table des matières

Chapitre I

Préambule

Chapitre II

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Si aucune information contraire n'est explicitement présente, vous devez partir du principe que les versions des langages et framework utilisés sont les suivantes (ou ultérieures) :
  - `Python 3`
  - `HTML5`
  - `CSS 3`
  - `Javascript EM6`
  - `Django 1.9`
  - `psycopg2 2.6`
- Sauf indication contraire dans le sujet, les fichiers en python de chaque exercice sur `Python` seul (d01, d02 et d03) doivent comporter à leur fin un bloc `if __name__ == '__main__'`: afin d'y insérer le point d'entrée dans le cas d'un programme, ou des tests dans le cas d'un module.
- Sauf indication contraire dans le sujet, chaque exercice des journées portant sur `Django` aura sa propre application dans le projet à rendre pour des raisons pédagogiques.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices : seul le travail présent sur votre dépot GIT sera évalué en soutenance.
- Vos exercices seront évalués par vos camarades de piscine.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.
- Sauf indication contraire dans le sujet vous ne devez pas inclure dans votre rendu :

  - Les dossiers `__pycache__`.
  - Les éventuelles migrations.
    Attention, il vous est tout de même conseillé de rendre le fichier `migrations/__init__.py`, il n'est pas nécessaire mais simplifie la construction des migrations.
    Ne pas ajouter ce fichier n'invalidera pas votre rendu mais vous *devez* être capables de gérer vos migrations en correction dans ce cas.
  - Le dossier créé par la commande `collectstatic` de `manage.py` (avec pour chemin la valeur de la variable `STATIC_ROOT`).
  - Les fichier en bytecode Python (Les fichiers avec une extension en `.pyc`).
  - Les fichiers de base de donnée (notamment avec `sqlite`).
  - Tout fichier ou dossier devant ou pouvant être créé par le comportement normal du travail rendu.

  Il vous est recommandé de modifier votre `.gitignore` afin d'éviter les accidents.

- Lorsque vous devez obtenir une sortie précise dans vos programmes, il est bien entendu interdit d'afficher une sortie précalculée au lieu de réaliser l'exercice correctement.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Par pitié, par Thor et par Odin ! Réfléchissez nom d'une pipe !

Chapitre III

Règles spécifiques de la journée

- L'interprète à utiliser est `python3`.
- Chaque exercice est indépendant. Si parmi les features demandées, certaines ont déjà été réalisées dans les exercices précédents, dupliquez- les dans l'exercice courant.
- Vous devez travailler dans une base de données `postgresql` nommée `formationdjango` et créer un role nommé `djangouser`, dont le mot de passe est `"secret"`, qui aura tous les droits dessus.
- Votre dossier de rendu doit être un projet Django. Le nom du projet doit être celui de la journée en cours.
- Nous allons utiliser le concept d'application de Django pour séparer les exercices : Chaque exercice de la journée doit se trouver dans une application Django distincte portant le nom de l'exercice correspondant et se trouvant à la racine du dossier de rendu, .
- Le projet Django doit être configuré correctement afin de remplir les conditions requises par les exercices. Aucune modification des configurations ne sera permise en soutenance.
- Vous ne devrez rendre aucune migration avec votre travail.
- Dans chaque exercice dont le cartouche mentionne `ORM`, vous devez exploiter l'ORM de `Django`. Aucune ligne de `SQL` ne doit être écrite.
- Dans chaque exercice dont le cartouche mentionne `SQL`, vous devez utiliser la librairie `psycopg2` et effectuer toutes les requètes en `SQL`.

Voici un exemple de structure typique pour un rendu de l'étudiant krichard, concernant la journée d42 et comprenant deux exercices :

```
|-- krichard
|   |-- .
|   |-- ..
|   |-- .git
|   |-- .gitignore
|   |-- d42
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   |-- ex00
|   |   |-- admin.py
|   |   |-- apps.py
|   |   |-- forms.py
|   |   |-- __init__.py
|   |   |-- models.py
|   |   |-- tests.py
|   |   |-- urls.py
|   |   |-- views.py
|   |-- ex01
|   |   |-- admin.py
|   |   |-- apps.py
|   |   |-- forms.py
|   |   |-- __init__.py
|   |   |-- models.py
|   |   |-- tests.py
|   |   |-- urls.py
|   |   |-- views.py
|   |-- manage.py
```

.

Soyez malin : factorisez votre code et rendez le facile à réutiliser, vous gagnerez du temps.

Chapitre IV

Exercice 00

	Exercice : 00
Exercice 00 : SQL - construction d'une table	
Dossier de rendu : $ex00/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une application Django nommée `ex00` ainsi qu'une vue à l'interieur de celle-ci qui doit être accessible via l'url : `127.0.0.1:8000/ex00/init`.

Cette vue doit créer une table SQL dans Postgresql à l'aide de la librairie `psycopg2` et retourner une page contenant "OK" en cas de succes, ou dans le cas contraire un message d'erreur décrivant le problème rencontré.

La table SQL doit répondre a cette description. :

- Elle doit se nommer `ex00_movies`.
- Elle ne doit être créée que si elle n'existe pas déjà.
- Elle doit contenir uniquement les champs suivants :
  - `title` : chaine de charactere variable, unique, d'une taille maximale de 64 octets, non nul.
  - `episode_nb` : entier, PRIMARY KEY.
  - `opening_crawl` : texte, peut être nul, pas de taille limite.
  - `director` : chaîne de caractères variable, non nul, d'une taille maximale de 32 octets.
  - `producer` : chaîne de caractères variable, non nul, d'une taille maximale de 128 octets.
  - `release_date` : date (sans l'heure), non nul.

Chapitre V

Exercice 01

	Exercice : 01
Exercice 01 : ORM - construction d'une table	
Dossier de rendu : $ex01/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une application nommée `ex01`, et dans celle-ci un modèle `Django` nommé `Movies` ayant exactement ces champs :

- `title` : chaine de character, unique, d'une taille maximale de 64 octets.
- `episode_nb` : entier, PRIMARY KEY.
- `opening_crawl` : texte - peut etre nul.
- `director` : chaîne de caractères d'une taille maximale de 32 octets, non nul.
- `producer` : chaîne de caractères d'une taille maximale de 128 octets, non nul.
- `release_date` : date (sans l'heure), non nul.

Ce modèle doit également redéfinir la methode `__str__` afin que celle-ci renvoie l'attribut `title`

Chapitre VI

Exercice 02

	Exercice : 02
Exercice 02 : SQL - insertion de données	
Dossier de rendu : $ex02/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Vous devez créer une application Django nommée `ex02`, et dans celle-ci, les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex02/init` : doit créer une table qui doit avoir en tout point les mêmes caracteristiques que celle demandées pour l'`ex00` à la différence près qu'elle se nommera `ex02_movies`.

  Elle doit retourner une page affichant "OK" en cas de succès, ou dans le cas contraire un message d'erreur décrivant le problème rencontré.

- `127.0.0.1:8000/ex02/populate` : doit insérer dans la table crée par la vue précédente, les données suivantes :

  - episode_nb : 1 - title : The Phantom Menace - director : George Lucas - producer : Rick McCallum - release_date : 1999-05-19
  - episode_nb : 2 - title : Attack of the Clones - director : George Lucas - producer : Rick McCallum - release_date : 2002-05-16
  - episode_nb : 3 - title : Revenge of the Sith - director : George Lucas - producer : Rick McCallum - release_date : 2005-05-19
  - episode_nb : 4 - title : A New Hope - director : George Lucas - producer : Gary Kurtz, Rick McCallum - release_date : 1977-05-25
  - episode_nb : 5 - title : The Empire Strikes Back - director : Irvin Kershner - producer : Gary Kutz, Rick McCallum - release_date : 1980-05-17

- episode_nb : 6 - title : Return of the Jedi - director : Richard Marquand - producer : Howard G. Kazanjian, George Lucas, Rick McCallum - release_date : 1983-05-25
- episode_nb : 7 - title : The Force Awakens - director : J. J. Abrams - producer : Kathleen Kennedy, J. J. Abrams, Bryan Burk - release_date : 2015-12-11

Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le nom du film, suivi du problème rencontré.

- `127.0.0.1:8000/ex02/display` : doit afficher l'intégralité des données contenues dans la table `ex02_movies` dans un tableau `HTML` y compris les éventuels champs nuls.

  Si aucune donnée n'est disponible, ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

Chapitre VII

Exercice 03

	Exercice : 03
Exercice 03 : ORM - insertion de données	
Dossier de rendu : $ex03/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une nouvelle app Django nommé `ex03` et créez dans celle-ci un modèle Django en tout point similaire à celui de l'`ex01`.

Cette app doit contenir les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex03/populate` : doit insérer, dans la table crée par la vue précédente, les données suivantes :
  - episode_nb : 1 - title : The Phantom Menace - director : George Lucas - producer : Rick McCallum - release_date : 1999-05-19
  - episode_nb : 2 - title : Attack of the Clones - director : George Lucas - producer : Rick McCallum - release_date : 2002-05-16
  - episode_nb : 3 - title : Revenge of the Sith - director : George Lucas - producer : Rick McCallum - release_date : 2005-05-19
  - episode_nb : 4 - title : A New Hope - director : George Lucas - producer : Gary Kurtz, Rick McCallum - release_date : 1977-05-25
  - episode_nb : 5 - title : The Empire Strikes Back - director : Irvin Kershner - producer : Gary Kutz, Rick McCallum - release_date : 1980-05-17
  - episode_nb : 6 - title : Return of the Jedi - director : Richard Marquand - producer : Howard G. Kazanjian, George Lucas, Rick McCallum - release_date : 1983-05-25

  - episode_nb : 7 - title : The Force Awakens - director : J. J. Abrams - producer : Kathleen Kennedy, J. J. Abrams, Bryan Burk - release_date : 2015-12-11

  Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

- `127.0.0.1:8000/ex03/display` : doit retourner une page `HTML` affichant l'intégralité des données contenues dans la table `Movies`, formatée dans un tableau `HTML`, y compris les éventuels champs nuls.

  Si aucune donnée n'est disponible, la page doit simplement afficher `"No data available"`.

`En soutenance, la migration sera effectuée avant les tests.`

Chapitre VIII

Exercice 04

	Exercice : 04
Exercice 04 : SQL - suppression de donnée	
Dossier de rendu : $ex04/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une app nommée `ex04`. Elle doit contenir les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex04/init` : doit créer une table qui doit avoir en tout point les mêmes caractéristiques que celle demandées pour l'app l'`ex00` à la différence près qu'elle se nommera `ex04_movies`.

  Elle doit retourner une page affichant "OK" en cas de succès, ou dans le cas contraire un message d'erreur décrivant le problème rencontré.

- `127.0.0.1:8000/ex04/populate` : doit insérer, dans la table crée par la vue précédente, les données décrites dans l'exercice `ex02`.

  Cette vue doit impérativement réinsérer toute donnée qui aurait été supprimée.
  Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

- `127.0.0.1:8000/ex04/display` : doit afficher l'intégralité des données contenue dans la table `ex04_movies` dans un tableau `HTML`, y compris les éventuels champs nuls.

  Si aucune donnée n'est disponible, ou en cas d'erreur la page doit simplement afficher `"No data available"`.

- `127.0.0.1:8000/ex04/remove` : doit afficher un formulaire `HTML` contenant une liste déroulante de titres de films et un bouton `submit` nommé `remove`.

Les titres de films sont ceux contenus dans la table `ex04_movies`.

Lorsque le formulaire est validé, le film selectionné est supprimé de la base de données et le formulaire est réaffiché avec la liste de films restants mise á jour.

Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

Chapitre IX

Exercice 05

	Exercice : 05
Exercice 05 : ORM - suppression de données	
Dossier de rendu : $ex05/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une nouvelle app Django nommé `ex05` et créez dans celle-ci un modèle en tout point similaire à celui de l'ex01.

Cette app doit contenir les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex05/populate` : doit insérer, dans le modèle crée pour cette application, les données décrites dans l'exercice `ex03`.

  Cette vue doit impérativement réinsérer toute donnée qui aurait été supprimée. Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

- `127.0.0.1:8000/ex05/display` : doit afficher l'intégralité des données contenues dans la table `Movies` de cette app dans un tableau `HTML` ; y compris les éventuels champs nuls.

  Si aucune donnée n'est disponible, ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

- `127.0.0.1:8000/ex05/remove` : doit afficher un formulaire `HTML` contenant une liste déroulante de titres de films et un bouton `submit` nommé `remove`.

  Les titres de films sont ceux contenus dans le modèle `Movies` de cette application.

  Lorsque le formulaire est validé, le film selectionné est supprimé de la base de données et le formulaire est réaffiché avec la liste de films restants mise á jour.

Si aucune donnée n'est disponible, la page doit simplement afficher `"No data available"`.

`En soutenance, la migration sera effectuée avant les tests.`

Chapitre X

Exercice 06

	Exercice : 06
Exercice 06 : SQL - Update d'une donnée	
Dossier de rendu : $ex06/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une nouvelle app Django nommé `ex06`. Elle doit contenir les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex06/init` : doit créer une table qui sera exactement la même qu'à l'exercice 00 à la différence près qu'elle se nommera `ex06_movies` et qu'elle disposera des champs supplementaires suivants :
  - `created` de type datetime (date et heure), qui, á la création, doit être automatiquement assigné á la date et heure actuelle.
  - `updated` de type datetime (date et heure), qui, á la création, doit être automatiquement assigné á la date et heure actuelle et automatiquement updaté à chaque mise à jour à l'aide du trigger suivant :

```
CREATE OR REPLACE FUNCTION update_changetimestamp_column()
RETURNS TRIGGER AS $$
BEGIN
NEW.updated = now();
NEW.created = OLD.created;
RETURN NEW;
END;
$$ language 'plpgsql';
CREATE TRIGGER update_films_changetimestamp BEFORE UPDATE
ON ex06_movies FOR EACH ROW EXECUTE PROCEDURE
update_changetimestamp_column();
```

- `127.0.0.1:8000/ex06/populate` : doit peupler la table créée par la vue précédente avec les données décrites dans l'exercice `ex02`.

  Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès,

ou dans le cas contraire : un message décrivant le problème rencontré.

- `127.0.0.1:8000/ex06/display` : doit afficher l'intégralité des données contenues dans la table `ex06_movies` dans un tableau `HTML`.

  Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

- `127.0.0.1:8000/ex06/update` : doit gérer l'envoi et la reception d'un formulaire. Ce dernier doit permettre de choisir un film dans une liste déroulante contenant les titres des films de la table `ex06_movies`, et d'écrire du texte dans un deuxème champ. En validant le formulaire, la vue doit remplacer, dans la table `ex06_movies`, le champ `crawling_text` du film sélectionné par le texte entré dans le formulaire.

  Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

Chapitre XI

Exercice 07

	Exercice : 07
Exercice 07 : ORM - Update d'une donnée	
Dossier de rendu : $ex07/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une nouvelle app Django nommé `ex07` et créez dans celle-ci un modèle en tout point similaire à celui de l'ex01, à la différence près que vous devez y rajouter les deux champs supplémentaires suivants :

- `created` de type datetime (date et heure), qui, á la création, doit être automatiquement assigné á la date et heure actuelle.
- `updated` de type datetime (date et heure), qui, á la création, doit être automatiquement assigné á la date et heure actuelle et automatiquement updaté à chaque mise à jour.

Cette app doit contenir les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex07/populate` : peuple le modèle crée précédemment avec les mêmes données qu'à l'`ex02`

  Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontrés.

- `127.0.0.1:8000/ex07/display` : affiche l'intégralité des données contenues dans la table `Movies` dans un tableau `HTML`.

  Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

- `127.0.0.1:8000/ex07/update` : doit gérer l'envoie et la reception d'un formulaire. Ce dernier doit permettre de choisir un film dans une liste déroulante contenant

les titres des films contenu dans le modèle `Movies` de cette application, et d'écrire du texte dans un deuxème champs.

En validant le formulaire, la vue doit modifier dans le modèle `Movies` de cette application, le champs `crawling_text` du film sélectionné avec le texte entré dans le formulaire.

Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

`En soutenance, la migration sera effectuée avant les tests.`

Chapitre XII

Exercice 08

	Exercice : 08
Exercice 08 : SQL - Foreign Key	
Dossier de rendu : $ex08/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une nouvelle app Django nommé `ex08`. Cette app doit contenir les vues accessibles via les urls suivantes :

- `127.0.0.1:8000/ex08/init` : Doit créer deux tables.
  La première doit se nommer `ex08_planets` et avoir les champs suivants :
  - `id` : serial, primary key
  - `name` : chaîne de caratères variable, unique, non null, d'une taille maximale de 64.
  - `climate` : chaîne de caratères variable.
  - `diameter` : entier.
  - `orbital_period` : entier.
  - `population` : gros entier.
  - `rotation_period` : entier.
  - `surface_water` : réel.
  - `terrain` : chaîne de caratères variable, d'une taille maximale de 128.

  La deuxième doit se nommer `ex08_people` et avoir les champs suivants :
  - `id` : serial, primary key.
  - `name` : chaîne de caratères variable, d'une taille maximale de 64.

- `birth_year` : chaîne de caratères variable, d'une taille maximale de 32.
- `gender` : chaîne de caratères variable, d'une taille maximale de 32.
- `eye_color` : chaîne de caratères variable, d'une taille maximale de 32.
- `hair_color` : chaîne de caratères variable, d'une taille maximale de 32.
- `height` : entier.
- `mass` : réel.
- `homeworld` : chaîne de caratères variable, d'une taille maximale de 64, foreign key, référençant la colonne `name` de la table `08_planets`

- `127.0.0.1:8000/ex08/populate` : Doit peupler les deux tables en copiant le contenu des fichiers fournis `people.csv` et `planets.csv` dans les tables correspondantes, respectivement : `ex08_people` et `ex08_planets`.
  Cette vue doit retourner une page affichant pour chaque fichier "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.
- `127.0.0.1:8000/ex08/display` : affiche tous les noms de personnages, leur planète d'origine ainsi que le climat, dont ledit climat est tout ou partie venteux ('`windy`'), trié par ordre alphabetique des noms de personnage.

  Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher `"No data available"`.

`Renseignez vous sur la methode copy_from de psycopg2`

Chapitre XIII

Exercice 09

	Exercice : 09
Exercice 09 : ORM - Foreign Key	
Dossier de rendu : $ex09/$	
Fichiers à rendre :	
Fonctions Autorisées :	
Remarques : `n/a`	

Creez une nouvelle app Django nommé `ex09` et créez dans celle-ci deux modèles. Le premier modèle que vous devez créer doit se nommer `Planets` et doit contenir les champs suivant :

- `name` : chaîne de caractères, unique, non null,. maximum 64
- `climate` : chaîne de caractères.
- `diameter` : entier.
- `orbital_period` : entier.
- `population` : gros entier.
- `rotation_period` : entier.
- `surface_water` : réel.
- `terrain` : chaîne de caractères.

Ce modèle doit également redéfinir la methode `__str__()` afin que celle-ci renvoie l'attribut `name`.

Le deuxième modèles que vous devez créer doit se nommer `People` et doit contenir les champs suivant :

- `name` : chaîne de caractères, max 64.

- `birth_year` : chaîne de caractères, max 32.
- `gender` : chaîne de caractères, max 32.
- `eye_color` : chaîne de caractères, max 32.
- `hair_color` : chaîne de caractères, max 32.
- `height` : entier.
- `mass` : réel.
- `homeworld` : chaîne de caractères, max 64, foreign key référençant la colonne `name` de la table `Planets` de cette application.

Ce modèle doit également redéfinir la methode `__str__()` afin que celle-ci renvoie l'attribut `name`.

Créez également dans cette app, une vue qui doit être accessible à l'adresse `127.0.0.1:8000/ex09/display`.
Cette vue doit afficher dans un tableau `HTML` tous les noms de personnages, leur planète d'origine ainsi que le climat, dont ledit climat est tout ou partie venteux ('`windy`'), trié par ordre alphabetique des noms de personnages.

Si aucune donnée n'est disponible, la vue doit afficher le texte `"No data available, please use the following command line before use:"` suivi d'une ligne de commande.

Cette ligne de commande doit être celle à exécuter depuis la racine de votre rendu afin d'insérer dans les modèles crées précédement, toutes les données présentes dans le fichier `ex09_initial_data.json` (fourni dans les ressources de la journée).

Vous devrez donc fournir ce fichier avec votre rendu.

`En soutenance, la migration sera effectuée avant les tests.`

Chapitre XIV

Exercice 10

	Exercice : 10
Exercice 10 : ORM - Many to Many	
Dossier de rendu : $ex10/$	
Fichiers à rendre : `file_to_turn_in`	
Fonctions Autorisées : `functions_authorized`	
Remarques : `n/a`	

Creez une nouvelle app Django nommée `ex10` et créez dans celle-ci trois modèles :

- `Planets` et `People` : Ces deux modèles doivent être identiques en tout point à ceux de l'exercice `ex09`.
- `Movies` : Ce modèle doit être identique en tout point à celui de `ex01` à la différence près que vous devez y rajouter le champ `characters`.
  Ce champ est de type `"many to many"` avec le modèle `People` et il permet de répertorier tous les personnages présents dans un film et se trouvant dans la table `People`.

Les fixtures nécessaires au peuplement des modèles sont présents dans le fichier `ex10_initial_data.json` fournis parmis les ressources de la journée.

Vous devez également créer dans cette app la vue accessible via l'url `127.0.0.1:8000/ex10`. Celle-ci doit afficher un formulaire disposant de ces champs (chacuns obligatoire) :

- `Movies minimum release date` : date
- `Movies maximum release date` : date
- `Planet diameter greater than` : nombre
- `Character gender` : liste déroulante contenant les différentes valeurs disponibles dans le champs `gender` du modèle People. Il ne doit pas s'y trouver plusieurs fois la même valeur.

Une fois validée, la vue doit chercher, renvoyer et afficher le ou les résultats.

Un résultat est un personnages dont le genre correspond au champ `'character gender'`, avec un film dans lequel il est présent et dont la date de sortie se situe entre `Movies minimum release date` et `Movies maximum release date`, avec sa planète d'origine dont le diamètre est supérieur ou égale à `Planet diameter greater than`.

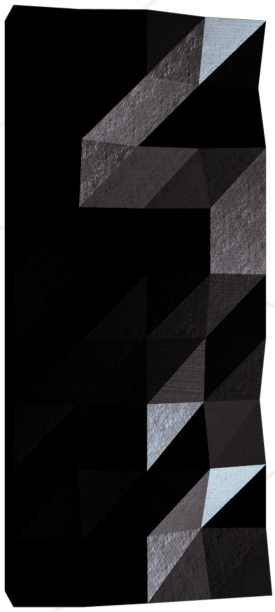
S'il n'existe aucun résultat correspondant à la requête, `"Nothing corresponding to your research"` doit s'afficher.
Chaque résultat doit s'afficher sur une ligne avec ces éléments (pas nécessairement dans cet ordre) :

- Le nom du personnage
- Son genre
- Le titre du film
- Le nom de sa planète d'origine (homeworld)
- Le diamètre de sa planete d'origine

Par exemple : les résultats pour des personnages de genre féminin, dont les films sont sortis entre "1900-01-01" et "2000-01-01" et dont la planète d'origine a un diamètre supérieur à 11000 sont :

- A New Hope - Leia Organa - female - Alderaan - 12500
- The Phantom Menace - Padmé Amidala - female - Naboo - 12120
- Return of the Jedi - Leia Organa - female - Alderaan - 12500
- Return of the Jedi - Mon Mothma - female - Chandrila - 13500
- The Empire Strikes Back -Leia Organa - female - Alderaan - 12500

Plusieurs personnages peuvent se trouver dans un même film, et un même personnage peut apparaître dans plusieurs films. C'est ce qu'on appelle une relation many to many (plusieurs à plusieurs). Dans ce cas, une table intermédiaire doit être crée entre ces deux tables. Chaque rang de cette table intermédiaire est une association (unique) de deux références : la première référençant un rang de la table des films, la deuxième référençant un rang de la table des personnages (ou dans l'ordre inverse). Une fois vos modèles construits et vos migrations effectuées, vous pourrez voir cette table via la console de postgre.



D06 - Authenti fi cation and users.

Site Project "Life Pro Tips".

Vincent Montécot vmonteco@student.42.fr

Summary: Where how to become a black belt permissions and user management with a simple Django project.

Contents

Chapter I

Preamble

Here are a few lines from a book talking artisanalle manufacture of IP addresses to the former:

I.1 TREATY From the art of making just hypées Address

As this will be found to written real art & infrequently exposed, its manufacture tion of hypées addresses, as the ancients taught in the workshops of the Great Vinton. Would to God that every one knew cestus just beautiful & much neglected subject, producing addresses if messy to use in trade in then.

Souventement we read and even manufacturing will practice both hated modern addresses. From the latter does not traictée ycelieu, for what it pro- few addresses Duit use, which wither and rot in no time as chiabrenas of virgins. Those explained in Vices pages haulte ma- SPECT and profound science, say real hypées are only able to withstand fl ancient scourge, Faillibus-aulniem, Altus-fi droopium epithelium. We must adoner songeusment both studie and under parfaicte workers in the preparation of ingredients and tools, easily save to pro- duce acute addresses, subtle, deep and seraines, and beautifully proportioned to achieve the highest degree exchanges.

I.1.1 The die beautifully aornées

As the faid saige Solomon, Wisdom points entrant in malivole soul, it is an important part of the influence of the manufacturer on manufactured like a mirror representing the most haulte virtue. So were it not that the strongest and healthiest workers who can complete that business and consume.

You can find the rest of this healthy reading right here .

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

Chapter III

Exercise 00

	Exercise :
00 Exercise: anonymous sessions.	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django, random, django-bootstrap3	
Remarks: n / A	

You start here a website project to provide users to post "Life Pro Tips".

After creating your project, an application and set up a homepage ac- transferable to the URL /, implement the following sessions of anonymous system:

- A user visiting your site should be assigned a username. This name is randomly selected from a list of 10 names that you must define in the fi le settings.py Your project has a duration of 42 seconds after its definition.

- This name must be persistent and remain the same for 42 seconds, once this period has elapsed, the name must be rede fi ned in the same way as before.

- Highlight the behavior and implemented by adding a message in a < nav> top of page. This message will be just as "Hello user" in remplacaçant wear by the name randomly assigned to visitors until the end of validity of the name in question.

- The name should appear immediately on the pages visited: If you need further update once the page so that it appears or is changed is that the exercise is not successful!

Chapter IV

Exercise 01

	Exercise :
01 Exercise: Creating users.	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django, django-bootstrap3	
Remarks: n / A	

Now that the system of anonymous session is added, it is time to implement user management with the creation of an authentication interface.

You will realize this interface in several stages in this exercise:

- Start by adding a registration page and a login page. Ajou- ter a link to each page in your element nav. Add to it the necessary URLs, templates for you, views, etc ... Add the site name at the beginning of your nav and make it a link to the home page.

- Create forms to insert in these pages with the specific following cities:
    - Registration Form :
        - You will need to insert three fields di ff erent including:
            - A field for the user name.
            - A field for the password.
            - A field for the verification password.
        - The validation shall 'E ff ectuer if every field is filled, the informed name does not already exist beforehand and the password fields

pass 2 well have identical values.

- Login Form:

  - You will need to insert two fields di ff erent including:
    - A field for the user name.
    - A field for the password.
  - The validation will not 'E ff ectuer if every field is filled, and the torque user / password filled exists in your database and allows an authentication.

- Now implement the behavior of pages:

  - If data validation:

    registration: Creation and activation of a new user with the infor-
    tions provided. Then connect to that user and redirect to the home page.

    Login: Login with the corresponding user information in-
    Trees, and then redirect to the home page.

  - In case of non-validation of the data:
    In all cases, you must first sheave ffi the page with the form containing the information previously entered (except passwords).

    registration: A ffi chage of one (or more) messages (s) explaining error
    the error (Required field, username already used, dif- ferent passwords)

    Login: A ffi chage an error message with the form (field re-
    quired or incorrect connection information).

- To fi ne when the user is logged:

  - links to registration and login pages are replaced with a logout link (obviously functional) that disconnects the visitors and redirects to the homepage once it clicked.

  - the name of the anonymous session in element nav is replaced with the name of the user.

  - visitors should no longer have access to the registration and login pages, if he tries to access it, it is redirected to the home page.

- Obviously, passwords should be hidden when they are typed.

In another exercise, you have to make your own user class to replace that provided by default. So you would do well (but not obliged) to anticipate this change in your project does not have to manually rewrite each call to your user class in your code.

Chapter V

Exercise 02

	Exercise: Exercise 02:
Our Tips!	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django, django-bootstrap3.	
Remarks: n / A	

It is time to implement the main feature of this project: A tuces as- system!

For this you will need to design a Model Tip with:

- A field contents ... with the contents of the tip (obviously as text).
- A field author (the user who created the tip).
- A field Date (the date and time of creation).

The tips will be listed on the home page and all the attributes mentioned in this statement should appear.

Once connected, the user must have the ability to create a Tip formulated with a lar on the home page (use ModelForm!)

If the user is not connected, the form should not appear and the visitor is not able to create tip.

Chapter VI

Exercise 03

	Exercise: Exercise
03: Votes	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django, django-bootstrap3.	
Remarks: n / A	

Tips Now that our system is in place, it would be nice to identify the best: It is time to put in place a system upvotes and downvotes!

For this, you will alter your Model Tip in order to implement these fonctionnali- tees.

You will also need to implement a way to remove the tips that we con- dérera not as interesting.

You need to add the tips listed on the home page:

- A delete button
- A button upvote
- A button downvote
- The number of upvotes
- The number of downvotes.

For now, the only restriction will implement the need to be connected to e ff ectuer shares voting or deletion. If the user is not logged in, none of these actions will be possible.

Chacunes these actions reload the page. Obviously, a Tip freshly created will upvotes 0 and 0

downvotes.

Warning :

- It should not be possible to upvoter and downvoter along the same tip.
- A user can upvoter or downvoter once one tip. Re-click on the button simply cancels the downvote or upvote.
- If a user downvote a tip auparavent upvote, the upvote is canceled in addition to downvote (and vice versa).

Pay particular attention to the implementation of your upvote system. An inconsistency, an imperfection (e.g., a number of votes not lowering after cancellation of the vote) invalidate exercise. Consider using ManyToMany fields rather than "counters" to implement the voting system.

At this point, everyone of action must only work if the visitor is logged.

You can not disable or a ffi expensive items to be unusable by the user.

Chapter VII

Exercise 04

	Exercise :
Exercise 04: First use of permissions.	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django, django-bootstrap3.	
Remarks: n / A	

Now that our project is a bit more complete we realize that anyone-can do anything provided it is connected. It is probably time to add some restrictions.

The user must now have permission to delete ... delete Tips for tips.

Enable the default administrative interface included with Django and use a su- perutilisateur to alter the permissions and thus highlight the behavior of your project.

Obviously, remove the "Remove" button on the page is not su ffi health (although you can). To you to find the right way to restrict this functionality by checking the permissions correctly.

Note: Obviously, the author of a tip can remove the tip even if he does not au torisation to remove in general. This is the only exception.

Chapter VIII

Exercise 05

	Exercise :
Exercise 05: custom permissions.	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django.	
Remarks: n / A	

Given the negative nature of the downvote, it would also restrict its use to certain well selected users.

Implement the same control as the previous year, but for downvotes. As such permission does not exist by default and it would be ugly that you read another UTI not adapted permission, make your own permission to restrict the use of the downvote!

Just like last year, he is not here simply to remove a page button.

Note: As with the removal of a tip, the author can downvoter own tips even if it is stupid.

Chapter IX

Exercise 06

	Exercise :
Exercise 06: Automation and reputation	
Render Folder: *ex/*	
Files to make: Your project, your file requirements.txt	
Permitted functions: All the features of Django, django-bootstrap3.	
Remarks: n / A	

Now we have most of the tools needed to run this project, it would be nice to not have to manually manage permissions. Some sites have implemented a gain privileges going with a reputation gain. This ma- Niere, users showing worthy unlock new actions! This is the subject of this exercise where you have to build a reputation system dependent on the votes received by Tips that a user has posted.

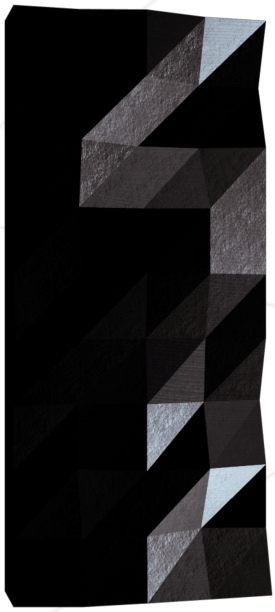
Replace the user class standard by your own user class. Implé- lying in this class a way to depend on the reputation of a user's permissions downvoter and remove tips. this user's reputation depend on itself the sum of upvotes and downvotes received by Tips posted by that user.

Each new user starts with 0 reputation points. Every upvote received on one of its pay him 5 Tips reputation points while a downvote him in- rise 2. If a tip is removed, the influence of his votes on the reputation of its author is canceled.

A user unlocks the permission of the Tips downvoter other than its own from 15 points of reputation, and that of removing from 30 to reputation. If a user's reputation down, it may lose the permissions that his reputation had made him.

Modi fi rm also has ffi chage the name of the user in a ffi singing beside his reputation

in your pages in parentheses.



D07 - Django-Python Training

Django - Advanced

Damien Cojan dcojan@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: Today we'll find some advanced features
Django.

Contents

Chapter I

Preamble

Happiness frameworks

I wanted to build a small shelf to store condiments. Having done some carpentry before, I had a good idea of what I be- care: some wood and some basic tools. A meter, a saw, a level and a hammer.

Besides, if I wanted to build a house, I need it also. So I went to a hardware store and I asked the seller where I could find a hammer.

"- A hammer?" He replied. "Nobody buys hammers nowadays you know. They are a bit old-fashioned. "

Surprised, I asked why.

- "Well, the problem with hammers is that there are lots of di ff erent kinds. nail puller hammers, sledge hammers, hammers upholsterer. . . What would happen if you were buying a type of hammer and realize that you need another type later? You should buy another hammer to your next task. It turns out that most people really want a single hammer that can be used for the majority of tasks they may face in their lives. "

- "It seems logical to me. Can you tell me where I can find a universal salt hammer? "

- "No, we do not sell them anymore. They are obsolete. "

- "Really ? I thought you just said that universal hammer was fu- ne. "

- "It turns out that if you make a single hammer that can be used for all kinds of tasks, it is not really good at any of them. Hammer a nail with a

mass is not very e ffi cient. And to kill your girlfriend, nothing beats an upholsterer's hammer. "

- "It's clear ! So if nobody buys Universal Hammers, and you sell more hammers old, what hammers do you sell? "

- "Actually, we do not sell."

- "So. . . "

- "According to our research, what people need is not a universal hammer at all. It is always better to have the right hammer for the good job. So we started selling hammer factories, capable of producing any hammer that may interest you. All you need is to fill the factory workers start machinery, buy raw materials, pay expenses and presto, you have * exactly * the kind of hammer you need in a flash. "

- "But I do not want to buy a factory hammers. . . "

- "Perfect. Because we sell more. "

- "Wait, you just said that. . . "

- "We found that most people do not need a full factory hammers. Some, for example, will never need an upholsterer's hammer. (Maybe they have no ex. Or maybe they killed them with ice picks.). So there is no reason for someone to buy a hammer factory for all types of hammers. "

- "Yes it's sure."

- "So, instead, we started selling the plans for construction of the factory hammers in order that our customers can build their own factories, com- pletely customized to produce only the types of hammers they need."

- "Let me guess. You sell them more. "

- "No. Of course. It turns out that people do not want to build a factory just to make some hammers. Let the construction of factories to factories construction experts, this is what I always say! "

- "And I agree with you on that."

- "And yes. So we stopped selling these plans and we started selling hammer factories factories. Each is built by our experts in the hammer factory factory business, in order that you do not have to worry about trivial details of the construction of a factory. Yet you

have all the bene fi ts of having your own customized factory producing your own custom hammers, sticking to your specific designs for hammer. "

- "Uh, it does not seem to me really. . . "

- "I know what you'll say! ! . . . and we sell more besides. Appar- ently, few people were buying the hammer factory factories, so we found a solution to this problem. "

- "Hmm."

- "We took the time to take stock of our technical infrastructure, we determined that people were developing a frustration with having to manage and operate a hammer factory factory, as it produced the factory. This kind of additional constraint can be tedious when you find yourself in a scenario where you also use a meter manufacturing factory, a saw mill and factory levels manufactures factory. Besides a conglomerate of wood processing. We have objectively assessed the situation and determined that it was too complex for someone who just wanted to create a shelf for spices. "

- "No kidding ?"

- "So this week, we bring to market a factory building factory manufactures tools of all kinds, whereby your di ff erent tools factory factories can be created from a single mill uni fi ed . The factory factory factory will produce only factory manufactures ment réelle- you need, and so makes these factories produce a single factory based on your custom tools speci fi cations. You will * exactly * the hammer you need, and exactly the right meter for your task, just by pressing a but- ton (though you'll probably few fi le configuration so that everything works as you expect).

- "So you do not have hammers? Not at all ?"

- "No. If you really want a shelf high quality condiments, industrial stan- dard, you really need something more sophisticated than a simple hammer bought at the hardware store. "

- "Okay. . . Good. It takes what it takes. If that's the way we do now, it is necessary that I put myself. "

- "Excellent! "

- "It comes with documentation, right?" Source: SametMax

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.

- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.

- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...

- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- During this pool of day you'll develop a unique site. It aims to offer items to view. It allows connected users to publish new articles or to save it as a favorite.
- You have the freedom to name this site as you please and to focus its thematic tick to your taste (news, fan fi ction, new erotic, etc ...).
- **You can choose the database you want, as long as it is compatible with the native ORM Django**
- **Your rendering will take the form of a single project Django. He will not have the usual cutting exercises.** Each rajoutent project functionality. It is this and its implementation that will be noted.
- **It is forbidden to implement any url "hard". You should always refer to name the url. Whether the views or** templates.
- **It is forbidden to implement any view as functions. You should use generic views.**
- Remember that the exercises will be corrected in order.
- The default language of your site is English. The content of the database can be anything. Latin if it suits you.
- You must leave the default management application.

Do not waste time on what is not asked!

Chapter IV

Exercise 00

	Exercise: 00
00 Exercise: Model building - Generic Class	
Render Folder: *ex* 00 / Files to	
make: Functions Authorized:	
Notes: n / A	

Create a new project. You can name it as you please and organize the structure of the application as you please. Think a logical structure and easy to understand facilitate corrections.

Implement the following models:

- **Article: Content articles and some metadata. It should contain the following fields:**
    - **title: Article Title. characters string with a maximum size of 64. No null.**
    - **author: Author of the article. Reference recording of the User model. No null.**
    - **created: Date and time of creation complete. Must be filled automatically with creation. No null.**
    - **synopsis: Article summary. characters string with a maximum size of 312. No null.**
    - **happy : The article itself. This is a text. No null. The __str () method must be __**

'override' to return 'title'

- UserFavouriteArticle: Items favorites recorded by a user. Ne must contained the following fields:
    - user: Reference recording of the User model. No null.
    - Article: Reference recording of the Article model. No null. The __str () method must be __ 'overriden' to return 'title' which is in the Article model.

Once done, we can move on to the real objective of this first exercise. Using only generic views (except 'View' you do not have permission to inherit directly), you must implement the following fonctionnali- ties into your site. Each of these features should have its own URL:

Articles: Page HTML a ffi song in table form HTML all fields (with the exception tion happy) of all items stored in the table Article.
The table must have a header indicating the title of each column.

Home: Url imposed: '127.0.0.1:8000. redirects to goods

Login: Page HTML a ffi singing a type of form POST. Logue non user logged with a user name and a password. On error, the page has a ffi expensive a message describing the error. On success, the view must redirect 'Home'.

You must also provide at least five articles of examples from three utilisa- ers di ff erent. Provide fi xtures if necessary. The content of the articles does not matter, the 'lorem ipsum' is perfect.

No css formatting is required as part of this exercise.

Chapter V

Exercise 01

	Activity: 01 year 01: Generic
Class again	
Render Folder: *ex* 01 / files to	
make: Functions Permitted	
Remarks: n / A	

Using only generic views (except 'View' you do not have permission to inherit directly), you must implement the following fonctionnali- ties into your site. Each of these features should have its own URL:

Publications: Page HTML a ffi song, in table form HTML the fields 'Title' 'synopsis' and 'Created' of all items stored in the model Article ' whose author is the currently logged on user.

For each item, you must also implement a link whose URL should contain the identifier of the said Article, leading functionality 'Detail' the ar- ticle concerned.

The table must have a 'header' indicating the title of each column.

Detail : Page HTML ffi song has any field of a given article being in base of data. The identi fi cation of this article must be in the url.

The field layout is free.

You must also add to each item in this table HTML functionality goods last year, a link to the 'Detail' the said Article.

Logout: Link déloguant a logged-in user. The place where is the link to logout does not matter in this exercise, as long as it is visible and accessible. Once logged out, the user is redirected to 'Home'

Favorites: page HTML a ffi song as a list of links, titles

Bookmark saved articles by the currently logged in user.

Each link, the url must contain the identifier of the item in question must carry out functionality 'Detail' the said Article.

You must provide at least one user having two favorites di ff erent.

Chapter VI

Exercise 02

	Exercise: 02
Exercise 02: Generic Class - CreateView	
Render Folder: *ex* 02 / Files to	
make: Functions Authorized:	
Notes: n / A	

Using only CreateView, you must implement the following functionalized bedded in your site. Each of these features should have its own URL:

Register: Page HTML containing a type of form 'POST' allowing a UTI
lisateur not logged create a new user account.

The form must request a login, a password and a confirmation password. This form must be accessible to a URL that is exclusively dedicated to him and that must end with 'Register'.

Publish: Page HTML containing a type of form 'POST' allowing a UTI
lisateur logged publish a new article. The 'author' must not be a ffi ket, it must be completed in the view during the validation process. You must use an object 'Form' created by your View to generate your form (not <input> Hand typed to the fields in your form!)

Add a link to this functionality in the template functionality Publications.

Add to favorite: Page HTML containing a type of form 'POST' is Trouville
efore in the detail page of an article. No field should be visible. Field Article ' must be pre-filled with the ID of the current article and upon validation, the field 'User' should be filled with the user ID logged. This mechanism Me- adds Article current favorites of the connected user.

No CSS formatting is required as part of this exercise.

You knew Django offers all ready forms?

Chapter VII

Exercise 03

	Exercise: 03
Exercise 03: Template Tags and Filters	
Render Folder: *ex* 03 / Files to	
make: Functions Authorized:	
Notes: n / A	

In this exercise you have to create a menu that is visible from ALL pages of the site.

All links lead to this menu of features YOU HAVE CREATED.
If you are missing something ... ask you questions.

This menu has the following items:

- **Home: Link functionality 'Home' (which in turn redirects to 'Articles'**
  you remember ?).
- **Last Articles: Link functionality Article. You can customize the name of this link according to any theme you**
  have chosen (but always in English!).

- If a user is not connected:
  - **Register: Link functionality 'Register'**
  - **Login: functionality 'Login'. Here is a little di ff erent because it is not a simple link, you need to put the**
    entire form IN the menu. It will be available on all pages, not just on the dedicated page that you have created.

    It must however always a ffi expensive mistakes if form inva-

lide.

- If a user is logged:
  - **Favorites: Link functionality 'Favorites'**
  - **Publications: Link functionality 'Publications'**
  - **Logged as <user>: Simple text indicating that the user is logged in. <User> must obviously be replaced** with the name of the connected user.
  - **Logout: functionality 'Logout'. Now you have a specific place to put this link.**

Using tags and filters, changed ez template listing all the items so that:

- The synopsis is reduced to a maximum of 20 characters, beyond which the following should be replaced by three dots. You must provide an example that is demonstrating.
- The list of items to be sorted by date, from most recent to oldest.
- An additional column mentions how long the article was published.

No CSS formatting is required as part of this exercise.

Chapter VIII

Exercise 04

	Exercise 04 Exercise 04:
Bootstrap	
Render Folder: *ex* 04 / Files to	
make: Functions Authorized:	
Notes: n / A	

Using Bootstrap, give to your CSS formatting the same menu as the one in the image provided in the resources of the day.

Chapter IX

Exercise 05

	Exercise 05 Exercise 05:
Internationalization	
Render Folder: *ex* 05 / Files to	
make: Functions Authorized:	
Notes: n / A	

Translate all functionality goods Site and the menu (which is visible from here also) in French based on the pre fi x lying in the URL.

For example: If my URL is: 127.0.0.1:8000/en/articles, then all its contents will be an- lish.

If this URL is 127.0.0.1:8000/fr/articles, then all its contents will be Fran- lish.

The content of the database and the site name does not translate. By default, the site must be in

English.
Add a link to switch from one language to another on the same page.

Chapter X

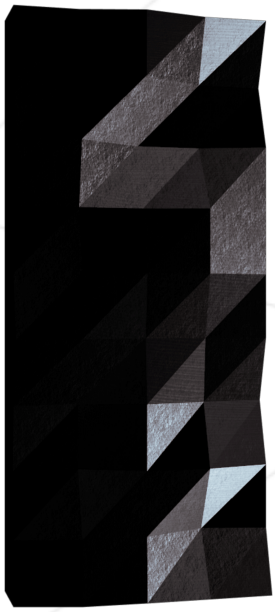
Exercise 06

	Exercise 06 Exercise
06: Testing	
Render Folder: *ex* 06 / Files to	
make: Functions Authorized:	
Notes: n / A	

Using the built-in test framework Django create tests to verify that the site out the following behaviors:

- The views favorites, publications and publish and their templates are accessible only to registered users.
- It is not possible for a user logged in to access the form for creating a new user.
- A user can not put the same article twice favorites. Your tests must imperatively wear a descriptive name, indicating exactly what they veri fi ed.

Correct any errors potentiellements revealed by such tests



D09 - Django-Python Training

Django - Ajax - Websockets

Damien Cojan dcojan@student.42.fr
42 Sta ff pedago@staff.42.fr

Summary: Today we'll find out how to use AJAX and Websockets with Django.

Contents

Chapter I

Preamble

Cat

The domestic cat (*Felis silvestris catus)* Following is the subspecies of the domestication of the wild cat, carnivorous mammal of the cat family. It is one of the leading pet and now has fifty races di ff erent re known by the bodies certi fi cation. In many countries, the cat is part of legislation on domestic carnivores like dogs and ferrets.

Source.

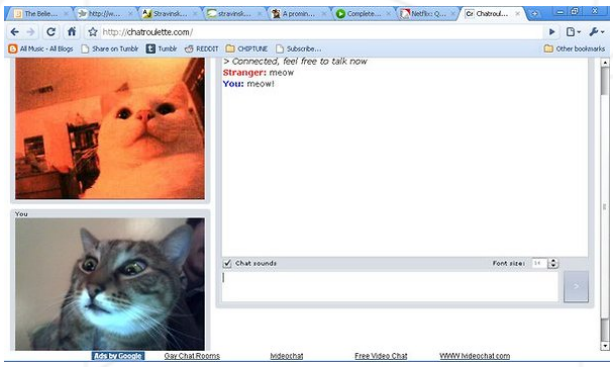


figure I.1 - Chat chatting on chatroulette.

No exercise during the day speaks of this cat there. I prefer to clarify before a doubt sets in. Do not think that I doubt your intelligence right? I prefer to just anticipate

. . . for those in the bottom near the radiator ... ... it's always those in the background anyway. * Sighs *

Chapter II

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

chapter III

Specific Rules of the day

- The only javascript library that you have to use is JQuery
- **Your rendering will take the form of a single project Django. He will not have the usual cutting exercises.** Each rajoutent project functionality. It is this and its implementation that will be noted.

- You must leave the default management application.
- You must make your project a file requirement.txt (via 'pip freeze') listing the libraries needed to run your project.

Chapter IV

Exercise 00

	Exercise: 00 00 Exercise: Ajax
my formulah!	
Render Folder: *ex* 00 / Files to make: The necessary files for your application	
Permitted functions: See setpoint	
Remarks: n / A	

Create a new project named d09 and in this project an application named account.

The objective of this exercise is to design a system connection / disconnection communicating only through AJAX.

You must implement this application in the url 127.0.0.1:8000/account which must return a page that may have two behaviors diff erent in two cases of FIG:

- The user is not connected: The page has a ffi expensive standard login form (login, password), except that communication with the server to validate the form should only use AJAX and must be of type POST.

  If the form is not valid, or the errors must be a ffi Chees on the page. If the form is valid, it must disappear

  and adopt another behavior. All this without the page is refreshed ... at any time.

- The user is already connected: The page has a ffi expensive the phrase " Logged as <user> "
  <User> is replaced by the name with which the user is connected, and a button Logout to log out.

This button should communicate with the server via AJAX and the method 'POST'. Once logged out, the sentence and the button must vanish from the page and it should adopt another behavior. All this without the page is refreshed at any time ... always

In case the page is refreshed manually ', this must regain the compor- ment where you left off (this does not include a ffi errors chage).

You can use Bootstrap.

AuthenticationForm is present!

Chapter V

Exercise 01

	Activity: 01 year 01: Basic
Chat	
Render Folder: *ex* 01 / files to make: The necessary files for your application	
Permitted functions: See setpoint	
Remarks: n / A	

Create a new application named 'cat'.

In this application you must create a page ffi singing three links that should lead to Cha- cun 'Chatroom' di ff erent.

The name of these rooms must be in the database, you will need to create a suitable model.

Each of these links should lead to another page containing a standard func- tional cat. Each cat must have the following characteristics:

- He must use 'JQuery' as single library and the frontend Websockets to communicate with the server. (No AJAX)
- It is only accessible to registered users.
- The cat's name must appear somewhere.
- Many users need to connect to it (*since once you guessed* ...) .
- It is possible for a user to post a message (*but you also know right?).*
- A message sent by a user must be visible to all other same chatroom (*everyone knows what a cat is not it? You read the preamble?).*

- Messages must ffi was expensive from top to bottom, from the oldest to newest (*this is for the original down to the bottom ... it's always those in the background anyway.)* accompanied by the name of the user who posted it.
- The messages should not disappear, a message should not replace another, the ordering of messages should not move.
- When a user logs, the message <Username> has joined the chat '
  should appear for all users including himself. <Username> is replaced by the name of the user.

Chapter VI

Exercise 02

	Exercise 02 Exercise 02:
History	
Render Folder: *ex* 02 / Files to make: The necessary files for your application	
Permitted functions: See setpoint	
Remarks: n / A	

In this exercise, you will improve your cat o ff him rant a history of mes- sages.

When a new user joins a chatroom, he must see ffi was expensive last three messages that have been posted this chatroom, from top to bottom, from the oldest to the newest.

Again only JQuery as frontend library and Websockets to communicate with the server are allowed.

Chapter VII

Exercise 03

	Exercise 03 Exercise
03: Userlist	
Render Folder: *ex* 03 / Files to make: The necessary files for your application	
Permitted functions: See setpoint	
Remarks: n / A	

In this exercise, you improve your cat o ff him rant a list of connected users that updates alone as much.

When the user connects to a chatroom, it must have access immediately to the list of connected users (including himself).

This user list must be separate visually the message list (another or other container html).

When a user connects to a chatroom, its name should appear in the list of other connected users.

When a user leaves a chatroom, its name must be removed from the list of other users connected and the message <Username> has left the cat ' ap- should appear (<username> to replace the name of the user in question) in response to messages posted.

Again only JQuery as frontend library and Websockets to communicate with the server are allowed.

Look primarily to build a functional logic. The goal is not optimization

.

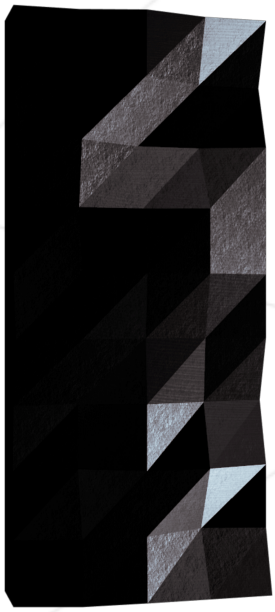
Chapter VIII

Exercise 04

	Exercise 04 Exercise
04: Scroll	
Render Folder: *ex* 04 / Files to make: The necessary files for your application	
Permitted functions: See setpoint	
Remarks: n / A	

Make your cat presentable by putting the list of messages in a container with a height and a fixed width. If the number of messages exceeds this height, the messages disappear too, and a scroll bar allows the scroll through.

In addition, the scroll bar should always be at the bottom, so that the latest messages are always in sight.



Python Django Training - Rush 00

Moviemon

Damien Cojan dcojan@student.42.fr

Summary: This is the subject of rush00 Python-Django pool.

Contents

Chapter I

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

chapter II

Specific Rules of the day

- The interpreter double use is python3
- The roads on an application must be defined in a file urls.py
  found in the record of this application.
- Each page has ffi chée must be properly formatted (presence of a doctype, tags couples html, body, head) Correct
  management of special characters, not a ffi weird chage.
- The server used for this rush is the default Django development server included with the utility manage.py.
- Only explicitly requested URLs must return an error page. Thus, if only / ex00 is requested / ex00foo should
  return a 404 error.
- You must make a file requirement.txt (via pip freeze) containing bookshops ries needed to run your project.

Chapter III

Preamble

Some quotes from the movie Be Kind Rewind with Jack Black (source):

figure III.1 - The video store.

Jerry: *I am Robocop. Anything you say can and will be Held against you in a court of Robocop.*

\- - - - - - - - -

Jerry: *I will shoot you, and I know karate robot.*

\- - - - - - - - - -

Jerry: *Saying I'm not your uncle is illerate, maybe he just needs to go to Nightschool.*

\- - - - - - - - - -

Jerry: *I Was going to make you into the next Marilyn Monroe! Goal! Now you're just going to be .... Laundry girl!*

\- - - - - - - - - -

Jerry: *Hey, hey, do not put your shoes in the refrigerator ... 'cause they'll get cold!*

- - - - - - - - -

Mike: (holding the Ghost Busters tape) *I'll be Bill Murray and you'll be everyone else.*

- - - - - - - - -

Jerry: [*sung, poorly, to the tune of the Ghostbusters theme song] When you're walkin 'down the street ...*

Jerry: [*singing] ... and you see a little ghost ...*

Jerry: [*singing] ... whatcha gonna do about -*

Jerry: *Ghostbusters?*

Mike: *What? What is that?*

Jerry: *That's the Ghostbusters theme song.*

Mike: *No.*

Jerry: *I'm pretty sure it is.*

- - - - - - - - -

Alma: *Are you in love with me?*

Mike: *Huh?*

Alma: *Mm-hm.*

Mike: *Well, how do I know that?*

Alma: *You know you're in love with a person When You talk to em for a minimum of 20 minutes a day in your head.*

Mike: *What if I talk to a guy in my head for 20 minutes? That What would mean?*

Alma: *You're in love with Jerry.*

- - - - - - - - -

Your name it, we shoot it

- - - - - - - - -

Sometimes the best movies are the ones we make up

chapter IV

mandatory part

IV.1 Introduction - FAQ

This rush to aim to make you encode a small single-player game with a web interface.

What is the purpose of this game?

This game is called MovieMon and its purpose is to capture all Moviemons hiding on a game board by using Movieballs.

That tells me something, it does not look like ...?

I do not know what you mean.

What is Moviemon?

A Moviemon is a film available on IMDB. Ideally a film Monster.

How do you get a Moviemon?

Throwing him a movieball see!. The IMDB rating of Moviemon match his strength. A high rating makes Moviemon more di ffi cult to catch a low rating. The strength of the player, which is the number of Moviemons in his possession, increases the chances of catching.

How is a typical new part?

When a player starts a new game, the game application on IMDB all films required before sending the 'Worldmap', the main page of the game. On this page, the player moves freely and cheerfully from box to box on a grid of fixed size. Random boxes on which he walks, he harvests movieballs, or uncovers a Moviemon.

When he uncovers a Moviemon, if it feels that its chances, the player tries to cap- turer.

It gives him a movieball, missed a second, missed a third, caught! The player then consults its fi proudly Moviedex which lists all Moviemons captured, before returning to hunt for all catch them! !

IV.2 instructions

IV.2.1 Settings

You must add to your project settings that of the game itself, namely:

- The size of the grid. This must be a minimum of ten squares tall and ten squares wide.
- The player's starting position in this grid
- The names or IDs of at least ten films on IMDB queryable. You must use these settings in your code. You can add other settings if you wish.

The rating of a film corresponds to its strength. So remember to balance your game by taking enough good and bad films. For example: at least three with score less than four score and three with more than seven

The API IMDB is rather obscure, you can use OMDB Which is an unofficial API.

IV.2.2 game data

You will need to retain data between di ff erent pages at a scéance game. A typical website would use cookies, or a system of server side ses- sion. But here there is no question of typical website.

You will need to store the data summarizing the state of the current game in a file to be created in your project.

This fi le is not a backup for the player he used to store the state of the current game. It should not contain any logic and its contents must be stored in binary thanks to the bookstore pickle (included with Python).

So you must also create your project in the logic needed to update and use of this fi le that should contain the following information:

- The player's position on the map.
- Number of Movieballs possessed.
- The names (or identifiers) of all Moviemons of Moviedex.
- Full details of all Moviemons of the game, as obtained on IMDB.

IV.2.3 Data Management

You must also create a class Python . Whose mission is to manage the dataset This class must contain at minimum the following methods:

- 'Load': load game data passed as parameters in the class instance. Returns the current instance.
- 'Dump': Returns the data set.
- 'Get_random_movie': Returns a random among the Moviemon Moviemons uncaptured.
- 'Load_default_settings': load game data in the class instance from the settings. Request and store the details of all Moviemons sure IMDB. Returns the current instance.
- 'Get_strength': Returns the strength of the player.
- 'Get_movie': Returns a dictionary Python containing all the details from the name of Moviemon passed as a parameter to make your page Detail.

You can add to this class all methods and attributes as necessary.

It will never be asked to change "by hand" or delete that file or any backup file during the game in defense.

IV.2.4 Aesthetics game

The a ffi chage the game will naturally on your browser via the HTML and CSS.
You do not have permission to use Javascript.

The a ffi chage of the game is divided into two parts which must be visually distinct perfectly:

- Screen: A ffi che what happens in the game. No interaction is possible at this point that should never contain any link or form.

- Controls : Located below or on either side of the screen, they can interact with the game and are contextual, ie they change compor- of charge depending on the state of the game.

  This also means that they are not necessarily all assets systematically. However, even inactive, they must be visible and keep the same place permanently.

  There must be eight 'buttons':

  - Four directions, such as one might find on a directional pad Joystick: Up, Right, Down, Left

  - A button select.
  - A button start.
  - A button AT
  - A button B

You do not have permission to add / remove 'buttons' nor a ffi expensive infor- mation, apart from the name of 'buttons' in this area.

Beyond this minimum distinction, aesthetics does not matter in the par- tie compulsory subject.

The behavior of buttons for each view is described in the next section.

A 'button' is not necessarily an HTML tag <> button. A 'button' inactive can be a dead link, an image, a text, or a special character.

IV.2.5 Pages

You must create pages / views / behaviors listed if after. A 'button' men- tioned not a page is a 'button' inactive. Moreover, if the destination is not specified for a check is that it returns the same page, potentially amended.

TitleScreen

- Description: Welcome Screen
- Display: Must a ffi expensive the name of the game and the A - New Game 'and' B - Load '.
- Url: the base, plus shipping domain.
- Controls:
  - AT : Link to the page Worldmap.
    Before a ffi expensive one, the fi le contains the information of the current part must be reset with the paramêtres Settings and the Moviemons
    must be queried again.
  - B: link to Load

worldmap

- Description: card game, where the character moves, recovers movieballs
  and flushes the Moviemons.
- Screen: A grid whose size is that defined in the settings. On the box corre- sponding to the current position of the player must find a representation (image, character, etc ...) clearly identi fi able character. The screen also has a ffi expensive:
  - Number of movieballs.
  - A message when movieball is found.
  - Message when Moviemon is flushed out and an indication of the but- ton to lean to begin capturing.]
- url: '/ Worldmap'
- controls:
  - directions: Each department must move the character one square in the same direction. The player must not get out of the map. Each movement has a chance to flush a Moviemon or to harvest movieball.

    If a Moviemon is flushed out, one is chosen at random from Moviemons

still not captured.

- **AT : Only if a Moviemon is flushed: page link Battle the fight against this Moviemon.**
- **start: link to Option.**
- **select: link to Moviedex.**

Refresh this page does not change the position of the character on the card.

Battle

- Description: Try to capture the Moviemon you flushed!
- **URL: '/ Battle / <moviemon_id>. < moviemon_id> is replaced by the ti fi er of identical Moviemon fight** whatsoever.
- **Screen: A ffi che poster and strength Moviemon, number of movieballs in stock, the strength of the player** and the chance of success rate (see below). If captured, you must also expensive ffi a sentence like "You catched it" to mark the event.

  On failure of a launch, you have a ffi expensive in the same way a phrase like "You missed!" ". As long as Moviemon is not captured, the screen also has a ffi expensive A - Launch movieball.

  In all cases, you need a ffi expensive than the 'button' B back to the Worldmap.
- controls:
  - **AT : Throws movieball**

    If the player does not, the action has no e ff ect (you can a ffi expensive on the screen a mockery of sentence from the enemy).

    Otherwise, the number of movieball is decreased by one and a lucky roll is made as to whether the Moviemon is captured or not. The chance to rate C is calculated as follows:

    C = 50 - (monster strength * 10) + (5 * player's strength) and 1 <= C <= 90

For example :
For a powerhouse and a force 8.2 Player 2: C = 50-82 + 10 = -22

This monster is thus 1% chance of having caught. For a powerhouse and a

force 5 player 8: C = 50-50 + 40 = 40

This monster is thus 40% chance of being caught. For a powerhouse and a

force 2 player 14 C = 50-20 + 70 = 100

This monster is thus 90% chance of being caught. If successful, the Moviemon is captured and stored in the MovieDex. The A button becomes inactive.

On failure, the player can raise as much movieballs it has.

- **B: Back to worldmap**

Moviedex

- **Description: List of Moviemons captured. It should be possible to select a film to access the details of it.**
- **URL: '/ Moviedex'**
- **Display: Must be present on the screen all the posters Moviemons captured. The AF selected fi che shall** be marked with a distinctive graphical element, for example by wrapping a blue border.

  You also add A - More Information 'and' select - Back '.
- controls:
  - **directions: Directions to select a film diff erent. You must use at least two directions: left and right or up** and down.
  - **select: Link to the page worldmap**
  - **AT : Link to the page Detail of Moviemon selected.**

Default arriving on the page, the first film in the list is selected.

Detail

- Description: Detail of a moviemon.
- URL: '/ Moviedex / <moviemon>' < moviemon> should be replaced by the identifier of the Moviemon.
- Display: Must a ffi dear name, the poster, the film director, year, rating, synopsis and players Moviemon and 'B - Back'
- Controls:
  - B Button: Link to the page Moviedex

Option

- Description: The game options.
- URL: '/ Options'
- Screen: A ffi che game options, which are A - Save ',' B - Quit 'and' start - cancel '
- controls:
  - Start Button: Link to the page Worldmap.
  - Button: Link to the page Save.
  - B Button: Link to the page TitleScreen.

Save

- Description: Allows you to save a game in progress in one of the three slots available.
- URL: / Options / save_game
- Screen: should a ffi expensive three slots: 'Slot A' 'Slot B' and 'C Slot.

  A selected slot shall be marked with a distinctive graphical element, for example by preceding it with an arrow.

  If a slot is empty, it must be followed by 'Free'. Otherwise, it must be followed by the number of Moviemons captured.

  For example: 'Slot A: 2/15' means that this backup contains a portion in which two Moviemons fifteen were captured. The screen also has a ffi expensive A - Save 'and' B - Cancel '
- controls:
  - directions: Directions 'high' and 'low' must be used to select a slot.

- AT : Copy the fi le contains the current state of the game in another fi le that must be stored in a folder 'saved_game' in your project. The name of the fi le must be 'slot <n> _ <score> .mmg. <N> is to replace with the name of slot 'a', 'b' or 'c' and <score> should be replaced by the score of the game.

  For example: slotb_1_15.mmg. (Please, do not try to put a slash in a fi le name).

- B: Link to the page Option

Load

- Description: Allows you to load a saved among three slots.
- URL: / Options / load_game
- Screen: should a ffi expensive three slots: 'Slot A' 'Slot B' and 'C Slot.

  A selected slot shall be marked with a distinctive graphical element, for example by preceding it with an arrow.

  If a slot is empty, it must be followed by 'Free'. Otherwise, it must be followed by the number of Moviemons captured.

  For example: 'Slot A: 2/15' means that this backup contains a portion in which two Moviemons fifteen were captured. The screen also has a ffi expensive A - Load 'and' B - Cancel '.

  Once a part is loaded A - Load 'should be replaced with' A - start game '

- directions: Directions 'high' and 'low' must be used to select a slot.
- AT :
  If no game is loaded. Copies the contents of the file corresponding to the slot selected in the fi le that stores the current state of the game if a part has been charged, the 'button' is used to link to the page Worldmap.

  Trying to load a slot 'Free' must obviously have no e ff ect.

- B: Link to the page TitleScreen

Chapter V Part

Bonus

Once your required part perfectly realized, you can implement additional functionality in order to earn bonus points.

For your checker sees these features as successful, you will have to convince their good achievement.

An unhandled error invalidate the functionality in question.

Only under the bonus, the use of Javascript is tolerated as long as the latter does not affect the operation of the compulsory part (which must work without JavaScript). The AJAX or Websockets are not allowed.

Here are some bonus ideas you could implement:

- Make your site look like a game console (see examples):
  - The screen should look like to be a real console screen.
  - The controls of your game to be superimposed on images of real controls mimicking a games console.
- In the current state of the game, it is perfectly possible to guess the address of a fight and catch the monsters without having to search. Create a system of token di ffi cult or impossible to guess, associated with Moviemons,

  to use in the URL in order to prevent cheating.
- Match the controls to the keyboard in order to avoid having to use the mouse to play.
- In order to vary the game, made that each new part load selected diff E- monsters pension.
- Add variety to the worldmap with a radar which has the ffi che Moviemons / moviballs
  available on adjacent squares to the player's position.

chapter VI

Rendering and peer-assessment

You must make a project Django perfectly configured.

Apart from what is imposed on you in the subject, you are free to organize the project as you wish.

No server error will be tolerated. Test out the behavior of your views. You must provide a file requirement.txt containing all the necessary libraries to run your project.

Django automatically returns a 404 page if the input URL was not found. You might come across cases where Django is a URL and receive, but your logic is wrong. In this case, consider using "raise Http404 ('my message')" in your views.

chapter VII

Example Rendering

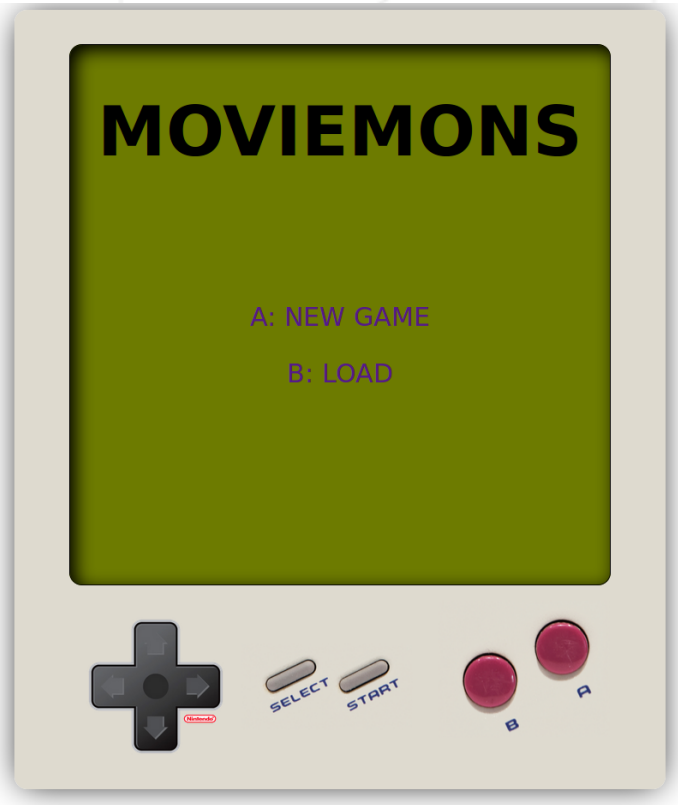
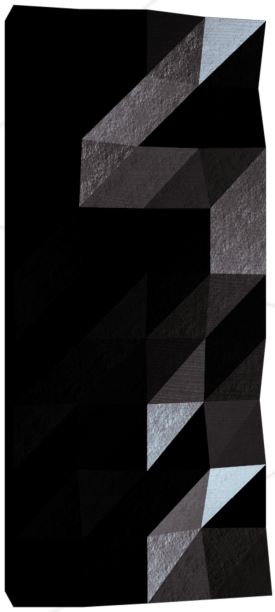


figure VII.1 - Your titlescreen Could look like that

figure VII.2 - This game Moviemon Appears to happen in Belgium



Rush01

Home stretch !

Vincent Montécot vmonteco@student.42.fr

Summary: It's time to rub in a real project!

Contents

Chapter I

Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you should assume that the versions of languages and frameworks used are (or later):
  - Python 3
  - HTML5
  - CSS 3
  - Javascript EM6
  - Django 1.9
  - psycopg2 2.6
- Unless otherwise indicated in the subject, the fi les in python each year on
  Python one (d01, d02 and d03) must have their end block if __name__ == '__main__': in order to insert the entry point in the case of a program, or tests in the case of a module.
- Unless otherwise indicated in the subject, each year days on
  Django will have its own application in the project to make for reasons peda- gogiques.
- The exercises are precisely ordered from simple to more complex. In no event shall we bear no attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- Unless otherwise specified in the topic you should not include in your rendering:

- The files __ pycache__.
- Any migration.
  Warning, you are still advised to make the fi le migration / __ init__.py,
  it is not necessary, but simplifies the construction of migration. Do not add this fi le will not invalidate your rendering but you *must* be able to manage migration for correction in this case.
- The file created by the command collectstatic of manage.py (with the way the value of the variable STATIC_ROOT).
- The fi le Python bytecode (fi les with extension. pyc).
- database fi les (especially with sqlite).
- Any fi le or folder must or can be created by the normal behavior of the rendering work.

It is recommended that you modify your. gitignore in order to avoid accidents.

- When you need to get an accurate output in your programs, it is of course forbidden to a ffi expensive output precalculated instead of performing the exercise cor- rectly.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that are not otherwise specified in the subject ...
- Please, by Thor and Odin! Re fl échissez dammit!

chapter II

Specific Rules of the day

- You can use it to this day-the following packages (over your own packages to build and install when correcting):

```
asgi-repeat == == asgiref
0.14.1 0.14.0 0.16.0 autobahn
== == channels 0.17.3 0.15.0
daphne == == Django 1.9.5


django-bootstrap3 == == 7.1.0 gunicorn
19.6.0 msgpack == python-0.4.8 Pillow
== == repeat 3.4.1 2.10.5 1.10.0 Twisted
six == == == 2.5 16.4.1 txaio .1 uwsgi ==
== 4.3.2 2.0.14 zope.interface
```

- You can also use JQuery and Nginx as you like.
- You are free to organize your project as you wish.
- Remember your fi le dependencies pip.
- When correcting the correction will completely alter the naviga- gateur side code to try to put in your rendering failure. You should take this into consideration to make your project su ffi ciently robust.

- Your rendering will be cloned with the command git clone <repo> ~ / rush01. So you have to set your project runs at that location.

Chapter III

Preamble

Here is a selection of Ig Nobel prize awarded in recent years:

2013 odds Price : Bert Tolkamp Mary Haskell, Fritha Langford, David Roberts and Colin Morgan, for discovering that the probability that a cow rises increases with the time during which it remains lying, and it is almost impossible to otherwise predict when it recouchera.

Prize for Peace in 2013 : Alexander Lukashenko, President of Belarus to making illegal applause in the public and the Belarusian police for arresting a unimanchiste accused of this crime.

2013 Psychology Price : Laurent Begue, Brad Bushman, Oulmann Zerhouni, Baptiste Subra and Medhi Ourabah to have experimentally confirmed that people thought to be drunk also think is attractive.

Physics Award 2014 : to Kiyoshi Mabuchi, Kensei Tanaka Daichi Uchijima and Rina Sakai studying the coe ffi cient of friction between soil, a banana peel and a shoe sole.

Neuroscience Prize 2014 : to Jiangang Liu Jun Li, Lu Feng, Ling Li, Tian Jie Kang and Lee, for trying to understand how some people see the face of Jesus in a piece of toast.

Physics Award 2015 : Patricia Hu Yang and David for showing that mammalian more than 3kg emptied their bladder 21s (+/- 13s).

2015 Biology Price : Bruno Grossi, Omar Larach Mauricio Canals Rodrigo A. Vásquez and José Iriarte-Díaz, having seen a chicken with a baton fi xed rump adopted a similar approach to the one that have probably had non-avian dinosaurs.

2015 Literature Prize : Mark Dingemanse Francisco Torreira and Nick J. Infi eld for discovering that the word "huh" existed anywhere without one really knows why.

chapter IV

mandatory part

You have arrived at this second rush that will close two weeks of swimming. This is an opportunity for you to demonstrate your skills with Django through the realization of an intranet.

IV.1 Base Project

You must first perform a basic project with Django with Sev- eral parts, among which:

IV.1.1 A con fi HTTP guration

A con fi guration server to serve your project with Nginx (You should *NOT* use the development server, DEBUG *MUST* be at False).

All features should work with this configuration. (If a function- ality works with ./ manage.py runserver but not with Nginx, this functional ality is invalid).

IV.1.2 An authentication system

Will only be available offline:

- A registration page.

  The form on this page must include a field for the user name and two fields for the password (which should not appear when typed). The validation will require that the user name is not already used and the values entered for the password are the same.

  Once the validation e ff ectuée, a user is created with the input and active data. The visitor is then connected with this user is redirected to the home page.

The password should obviously not be stored unencrypted in database.

- A login page.

  The form on this page must include a field for the user name and a field for the password (which should not appear when typed). The validation will require the user exists, and that the authentication with him is possible.

  Once the form is validated, the visitor is connected with this user is redirected to the home page.

  Once connected, the user must be redirected to the homepage. Once connected, these pages are no longer accessible. The user will then have access to a disconnect button.

IV.1.3 A homepage

A home page to access all the features that you can implement.

IV.1.4 Manage user profiles

You must associate each user some information among which:

- His name.
- His first name.
- His email.
- Its description.
- His picture of pro fi le.

You must provide on the page profile of each user the option (for this user and administrator) to view and edit the information.

Administrators must also be able to assign or remove a user administrative rights (It must be able to know whether a given user has administrative rights).

An administrator can very well be a superuser, a user with a set of permissions or part of a group. At your convenience.

IV.2 Functionality choice

Once this project is, you will implement at least one of the following features (if you do both you extra points).

IV.2.1 Forum

This forum, although simple, is to implement a system of posts. These posts will be listed by 10 on a page (bring up a paging system when the posts are too many).

Links to the detail of the listed posts will be present.

A post should have a title, author, content, date and time of creation and a set of comments that will be associated with it.

The comments have content, date and time of creation as well as an author.

The detail of a post a ffi chera all information of this post and all the comments (and their respective information as well).

It shall be possible to comment on a post as it will be possible to comment Comment (Your comment will be organized as a tree). The a ffi chage indi- Quera clearly what a comment is a comment (post or this or that comment).

All users must have the ability to create posts and comment.

IV.2.2 A messaging system

Your mail system will allow a user to hold conversations tion with other users by exchanging messages.

A conversation between two users will always *di ff erent (* And the content of that conversation is accessible only to those users only).

Your mail system will include two pages:

- A page listing conversations *started* involving the current user. You need a ffi expensive the last message of information received / sent (date / time, content truncated to 30 characters, party ...) for each conversation. A ffi in by 10 and bring up a paging system if the messages are too many to be a ffi kets on a single page.

also to implement each conversation listed a link to the detail of it.

Also add a link to start a conversation with a user *different* on his page of pro fi (If the conversation does not exist).

- The details of each conversation.

  You will have to list all the messages exchanged by chronological order with their contents and the date and time of issue. The a ffi chage must also clearly indicate whether a message was sent or received. The user with whom the conversation is held must also appear.

chapter V

optional part

Once the mandatory part performed, you can achieve bonus among the sui- lowing:

V.0.1 Bonus Forum: implementation of AJAX

Refresh this page should not be necessary for a ffi expensive a form for com- mentary, fill, submit it and see the changes was ffi expensive after validation by the server, and repeat.

Of course, this bonus is only feasible if you realized the forum.

V.0.2 messaging Bonus: use websockets

Using websockets in your messaging system:

The messages you are sent by another user should appear instantaneous tannément in the thread conversation without having to reload the page.

Of course, this bonus is only feasible if you have implemented a messaging system.

V.0.3 Push notifications

You must, using websockets, implement a system of push notifications, two cases may arise:

- You must show a notification when a new post is created if you did the week. This notification must at least bring up the post title truncated to 30 characters.

- You must show a notification when a new message is sent to you if you realized the mail system. You need at least a ffi expensive

the name of the sender and the message truncated to 30 characters. If you have validated both the messaging system and the forum, treat only one of the two cases su ffi t to validate this bonus.

V.0.4 SSL

Use your site to HTTPS and use your potential websockets in WSS with a key and an SSL certificate.

While accessing your site via HTTP, redirection to the HTTPS version is automatically e ff ectuée.

V.0.5 Package

Make one of your application as a package that will be built and installed with pip when correcting.

V.0.6 System votes

Implement a polling system to submit elements of your site users vote.

These votes will be positive votes (upvotes) or negative (downvotes). This feature will allow a ffi chage of some information:

- voting buttons (for downvoter and upvoter the element).
- A summary of the votes (positive and negative) for this element.
- The current vote of the user for this element (upvote, downvote or no vote) A user can vote only once on an item and can not simultane- ously upvoter and downvoter this element.

If the user clicks a second time on the same vote button, the previous vote is simply canceled.

If the user clicks an item for downvoter after upvote, downvote and the item is the previous upvote is canceled, the reverse is also true.

Once the click a button that is e ff ectué, information are updated on a ffi chage.

Once your done voting system, test it on your forum (On posts and com- ments). Check that having more element to be voted on the same

page does not disturb the di ff erent voting buttons, and each behaves cor- rectly (eg click a button to upvote not upvote all page elements.