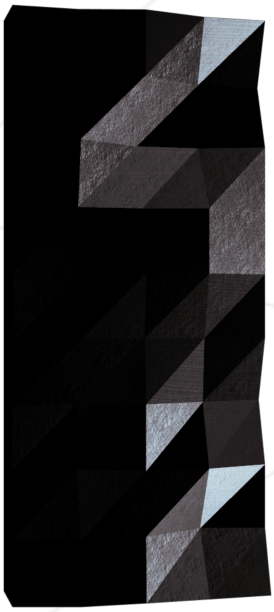

D01 - Django-Python Training

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> [...]
```