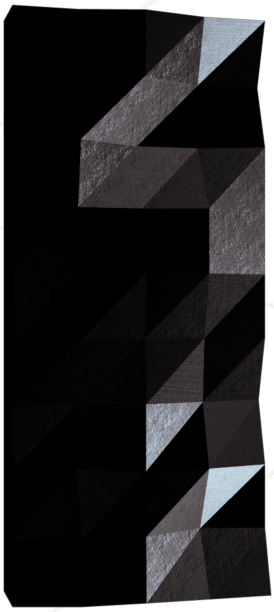

D01 - Training Ruby on Rails

syntactic and semantic bases

Stéphane Ballet balletstephane.pro@gmail.com
42 Sta ff pedago@staff.42.fr

Summary: We leave a little dommaine web to focus on the ruby <3, and the syntactic and semantic foundations of this language.

Contents

Chapter I

Preamble

Besides being awesome, Ruby is funny !

- Poignant Guide To Ruby
- TryRuby
- RubyMonk
- Rubyquizz
- RubyWarrior

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 must assume the following language versions:
  - Ruby 2.3.0
  - for d09 Rails> 5
  - but for all the other days rails 4.2.6
  - HTML5
  - CSS 3
- We prohibit EXPRESSLY using keywords while, for, redo, break, retry and until in Ruby source code that you will make. Any use of these keywords is considered cheating (and / or unfit), giving you the score of -42.
- The exercises are precisely ordered from simple to more complex. In any case, we do not bear 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.
- 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

not otherwise specified in the subject ...

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

chapter III

Specific Rules of the day

- All fi les will be returned with an appropriate shebang and warning fl ag.
- No code in the global scope. Make functions!
- Each fi le made must be completed by a function call.
- No authorized import, except for those explicitly mentioned in the "Permitted functions" of the cartridge each year.

chapter IV

00 Exercise: Class No Class

	Exercise: 00 00 Exercise:
Class No Class	
Render Folder: *ex* 00 / Files to	
make: var.rb	
Permitted functions: n / A	
Remarks: n / A	

Create a named script var.rb in which you must define a function my_var.
In it, declare and initialize variables 4 di ff erent types and print to standard output. You must reproduce exactly the following output:

```
$> Ruby var.rb my
variables:
        Contains a 10's type: Fixnum b Contains: 10 is of type String c
    Contains: ten of type String s Contains: 10.0 is type: Float $>
```

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.

chapter V

01 Exercise: Breakfast

	Exercise 01 Exercise 01:
Breakfast	
Render Folder: *ex* 01 / files to make: croissant.rb	
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 randomly separated by a comma.

Design a script Ruby appointed croissant.rb 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.

Command to extract a tarball: tar -xzf d01.tar.gz

chapter VI

Exercise 02: Hashment well

	Exercise 02 Exercise 02:
Hashment well	
Render Folder: *ex* 02 / Files to	
make: H2o.rb	
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 H2o.rb in which you must copy the table couples data as is in the next one or another of your duties:

```
data = [[ 'Caleb', 24],
                [ 'Calixte', 84], [ 'Tanager',
                65], [Calvin ', 12], [Cameron',
                54], [ 'Camil', 32], [ 'Camille',
                5], [ ' Can '



                                , 52],
                [ 'Caner', 56], [ 'Cantin', 4],
                [Carl '
                                , 1]
                [ 'Carlito', 23], [ 'Carlo', 19], [
              'Carlos', 26], [ 'Carter', 54], [
              'Casey', 2]]
```

Write code that, when it is executed, and declares the hash transforms with the key to the whole and as value the string and ffi song on the display a message as follows:

```
$> Ruby H2o.rb 24:
Caleb 84: Calixte 65: 12
Tanager Calvin 54:
Cameron 32: 5 Camil
Camille 52: Can 56:
Caner 4: Cantin 1: Carl
23: Carlito 19: Carlo 26:
Carlos 54: Carter 2:
Casey $>
```

chapter VII

Exercise 03: Where am I?

	Exercise: Exercise 03 03:
Where am I?	
Render Folder: *ex* 03 / Files to make: Where.rb	
Permitted functions: n / A	
Remarks: n / A	

Using the following hashes (you recopierez them in a function that either I do the most write):

```
states = {
    "Oregon"                => "OR",
    "Alabama"               => "G"
    "New Jersey" => "NJ", "Colorado"
  => "CO"}


capitals_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.

```
$> Ruby Where.rb Oregon Salem

$> Ruby foo Where.rb Unknown
state $> ruby Where.rb

$> Ruby Where.rb Oregon Alabama
$> Ruby Where.rb Oregon Alabama Ile-de-France $>
```

chapter VIII

Exercise 04: Backward

	Exercise 04 Exercise 04:
Backward	
Render Folder: *ex* 04 / Files to make: erehW.rb	
Permitted functions: n / A	
Remarks: n / A	

You have again the same two hashes than last year. Create a program that takes a capital argument and a 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.

```
$> Ruby erehW.rb Salem Oregon

$> Ruby foo erehW.rb Unknown
capital city $> $ erehW.rb ruby>
```

chapter IX

Exercise 05: Hal

	Exercise 05 Exercise
05: Hal	
Render Folder: *ex* 05 / Files to make: whereto.rb	
Permitted functions: n / A	
Remarks: n / A	

Always re-hashes of EX03, write a similar program to cedents pre- except that:

- The program should take as argument a string containing many words to search you want, separated by commas.
- For each word of that string, the program should detect if that word is a capital city or a state.
- The program should not be case sensitive, or white spaces.
- If there is no parameter or too many parameters, the program has nothing ffi che.
- When there are two commas in a ffi threaded into 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:

```
$> Ruby whereto.rb "Salem, Alabama, Toto,                    Montgomery "
Salem is the capital of Oregon (AKR: OR) Montgomery is the capital of
Alabama (AKR: AL) is toto Neither has capital city has state nor Montgomery is
the capital of Alabama (AKR: AL) $>
```

chapter X

Exercise 06: Wait a minute

	Exercise 06 Exercise 06:
Wait a minute	
Render Folder: *ex* 06 / Files to make: CoffeeCroissant.rb	
Permitted functions: n / A	
Remarks: n / A	

Using the following table:

```
data = [[ 'Frank', 33], [
'Stacy', 15], [ 'Juan', 24],
[ 'Dom', 32], [ 'Steve',
24], [ 'Jill', 24] ]
```

Write the code that ffi only che names sorted in order of increasing age and al phabétique when ages are identical, line by line.

```
$> Ruby CoffeeCroissant.rb Stacy Jill Dom
Juan Steve Frank $>
```

The hash of the data will be changed for the defense to verify that the work was done properly.

chapter XI

Exercise 07: elm

	Exercise 07 Exercise
07: elm	
Render Folder: *ex* 07 / Files to	
make: elm.rb	
Permitted functions: n / A	
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> [...]
```