



# Ruby on Rails Training - 1

## Gems

*Summary: You can always rely on a Gem. This module is achievable with Ruby version  $\geq 3$*

*Version: 1.1*

# Contents

<b>I</b>	<b>Preamble</b>	<b>2</b>
<b>II</b>	<b>General rules</b>	<b>3</b>
<b>III</b>	<b>Specific instructions for today</b>	<b>4</b>
<b>IV</b>	<b>Exercise 00: I like</b>	<b>5</b>
<b>V</b>	<b>Exercise 01: ft_wikipedia</b>	<b>7</b>
<b>VI</b>	<b>Exercise 02: TDD</b>	<b>9</b>
<b>VII</b>	<b>Exercise 03: Rails</b>	<b>10</b>
<b>VIII</b>	<b>Submission and peer-evaluation</b>	<b>11</b>

# Chapter I

## Preamble

The RS7000 is a major groove production workstation! It's sort of like Akai's MPC-series (but for bosses), combining sampling and sequencing, but with an added internal synth engine. The RS7000 is particularly suited for dance, techno, Hip Hop, R&B, and ambient genres.

The sampler section consists of a 4MB (expandable to 64MB) sampler (5kHz to 44.1kHz or 32kHz to 48kHz via digital option board). You can use it to sample external sounds, re-sample the RS7000's sounds itself, or load samples from a variety of common formats! Auto-beat slicing lets you easily sample any loops or sounds and sync them to your sequence tempo! All the professional sampling and editing features you'd expect are here, and more!

The tone generator offers 62-voice polyphonic AWM2 synthesis, with over 1,000 synth sounds and 63 drum kit sounds (all via ROM). Here you'll find the resonant filters (6 types), advanced LFO modulation, BPM-synchronized LFO waveforms, and more! Edits made to the internal sounds, as well as to any samples are all stored within your sequence patterns.

The Sequencer is the real meat of the RS7000, where you make music out of the sounds it's got and that you've put into it! It offers pattern-based recording with 16 tracks each, and a 200,000 note-per-song capacity. Linear sequencer sequencing, like you would do using a software sequencer like Cubase, is also supported by the RS7000. Pattern-based sequences can be converted to the linear format as well. Realtime, grid and step recording methods are also available. Linking patterns into songs can be done in real time and meticulously tweaked.

Total MIDI control, real-time hands on control, 18 assignable knobs and two pads, a Master effect section (with a multi-band compressor, slicer, isolater, other DJ-style master effects), and more make the RS7000 the most professional quality groove/loop/dance machine out there!

# Chapter II

## General rules

- Your project must be realized in a virtual machine.
- Your virtual machine must have all the necessary software to complete your project. These softwares must be configured and installed.
- You can choose the operating system to use for your virtual machine.
- You must be able to use your virtual machine from a cluster computer.
- You must use a shared folder between your virtual machine and your host machine.
- During your evaluations you will use this folder to share with your repository.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.


# Chapter III

## Specific instructions for today

- Every turned-in files will feature a fitting shebang AND the warning flag.
- No code in the global scope. Make functions or classes!
- 
- Each turned-in file must be a Gem. (except for exercise 03!)
- Each Gem must use `minitest`, the MIT license and must not propose any conduct code.
- The Gems have an educational aim: don't bother with the upload rests! **Just comment the corresponding lines in the .gemspec.**
- Each Gem must include the tests required in the exercise and they must be executed by a `bundle exec rake` in the Gem's root folder.
- Imports are prohibited except for the ones specified in the "Authorized functions" section in each exercise cart.

# Chapter IV

## Exercise 00: I like

	Exercise 00
Exercise 00: I like	
Turn-in directory : <i>ex00/</i>	
Files to turn in : <b>deephought</b>	
Allowed functions : <b>colorize</b>	

Create your first Gem!

By doing so, you will give birth to a reusable code component for the entire world. Providing the following code in a portable manner:

```
require 'colorize'

class Deepthought
  def initialize
    end
  def respond(question)
    if question == "The Ultimate Question of Life, the Universe and Everything"
      puts "42".green
      return "42"
    else
      puts "Mmmm i'm bored".red
      return "Mmmm i'm bored"
    end
  end
end
```


Exciting, right?

Your Gem must observe the following specifications:

- Gem name: `deephought`
- Tests? Of course! Use `minitest`
- Code of conduct? Not required.
- License: MIT
- Version: `'0.0.1'`
- The command `"grep -Hrn 'TODO' -color=always ."` executed at the root of your Gem, should not return any results.
- You must write a test to verify that `"Deephought.new"` returns the expected object.
- You must write a test to verify the return value of both `"respond"` methods.

# Chapter V

## Exercise 01: ft\_wikipedia

	Exercise 01
Exercise 01: ft_wikipedia	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <b>ft_wikipedia</b>	
Allowed functions : <b>nokogiri/open-uri</b>	

One interesting fact about Wikipedia is the road of philosophy. If you click on the first non-parenthesized, non-italicized link in 94% of cases, you will eventually reach the philosophy page.

Try it out yourself by creating a gem called "ft\_wikipedia" that can be used and displayed as follows:

```
Ft_wikipedia.search("Kiss")
First search @ :https://en.wikipedia.org/wiki/Kiss
https://en.wikipedia.org/wiki/Love
https://en.wikipedia.org/wiki/Affection
https://en.wikipedia.org/wiki/Disposition
https://en.wikipedia.org/wiki/Habit_(psychology)
https://en.wikipedia.org/wiki/Behavior
https://en.wikipedia.org/wiki/American_and_British_English_spelling_differences
https://en.wikipedia.org/wiki/English_orthography
https://en.wikipedia.org/wiki/Orthography
https://en.wikipedia.org/wiki/Convention_(norm)
https://en.wikipedia.org/wiki/Norm_(philosophy)
https://en.wikipedia.org/wiki/Sentence_(linguistics)
https://en.wikipedia.org/wiki/Word
https://en.wikipedia.org/wiki/Linguistics
https://en.wikipedia.org/wiki/Science
https://en.wikipedia.org/wiki/Knowledge
https://en.wikipedia.org/wiki/Awareness
https://en.wikipedia.org/wiki/Conscious
https://en.wikipedia.org/wiki/Quality_(philosophy)
https://en.wikipedia.org/wiki/Philosophy
=> 19
```





According to the changes made on Wikipedia between the time this example is created and the time you carry out the project, there may be a slight difference in the outcome.

The programs should lists all the visited URLs and returns the number of links it had to traverse before reaching the page "https://en.wikipedia.org/wiki/Philosophy".

Sometimes, searches like “matter” end up in a loop!  
You must handle this case by raising a “StandardError” exception and display the following:

```
Ft_wikipedia.search("matter")
First search @ :https://en.wikipedia.org/wiki/matter
https://en.wikipedia.org/wiki/Atom
https://en.wikipedia.org/wiki/Matter
https://en.wikipedia.org/wiki/Atom
Loop detected there is no way to philosophy here
=> nil
```

Sometimes, searches like "Effects\_of\_blue\_lights\_technology" lead to dead ends!  
You must handle this by raising a "StandardError" exception and display the follows:

```
Ft_wikipedia.search("Effects_of_blue_lights_technology")
First search @ :https://en.wikipedia.org/wiki/Effects_of_blue_lights_technology
Dead end page reached
=> nil
```


Also, Keep in mind that by “first link”, we mean a link available in the article that meets the following criteria:

- It is an article available on Wikipedia
- It is an article in the same language
- It is a legitimate article (not a file or help page)

You must also write all the tests that prove the proper functioning of your program with appropriate searches : "directory", "problem", "Einstein", "kiss", "matter" etc.  
These are just examples, and you are not bound to them

# Chapter VI

## Exercise 02: TDD

	Exercise 02
Exercise 02: TDD	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>Taillste</b>	
Allowed functions : <b>n/a</b>	

Test-Driven Development (TDD), is a widely practice approach in the world of web development.

We provide you with an empty gem that you need to implement. Test have already been written for it.

Based on these tests, it's up to you to determine what your gem should do.


To validate the exercise, it is imperative that :

- The “gem build” command at the root of your gem folder executes without error (except for help, missing description, and missing homepage warnings).
- The last line of the output when running the “rake” command should be:

```
"16 runs, 16 assertions, 0 failures, 0 errors, 0 skips"
```

# Chapter VII

## Exercise 03: Rails

	Exercise 03
Exercise 03: Rails	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <b>HelloWorld</b>	
Allowed functions : <b>n/a</b>	

Aaaah! Hello World! The famous one. So now you're facing the challenge.

You need to install **rails** on your virtual machine, create a page with a title "Hello World!", and when you launch the built-in server in Rails, this page should be the first one displayed.

Google is filled with good advice, but we have another one for you: document yourself before starting your installation...

It is crucial that your installation goes smoothly since you will have to do it multiple times. It's better to get off to a good start!.

To validate the exercise, you need to:

- Install the Rails gem, including the necessary dependencies.
- Ensure the built-in Rails server starts (the command is "rails server" or "rails s")
- Ensure that the first page of your website (accessible via "http://localhost:3000/") is an HTML page displaying "Hello World!" within an **<h1>** tag.

# Chapter VIII

## Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.



The evaluation process will happen on the computer of the evaluated group.