Table des matières

Documentation

http://www.tutorialspoint.com/ruby/index.htm

Astuces

```
#  Rend le script exécutable sous GNU/Linux
#!/usr/bin/env ruby
#  Sous Windows cette ligne est inutile, il faut associer les fichiers *.rb avec ruby.exe
#  pour pouvoir appeler le script sans le préfixer avec ruby : ruby mon_script.rb
```

```ruby
# le fichier doit être formaté en ANSI

# commentaire
=begin
      commentaire
      multi-lignes
=end

# test si un objet est null
if objet.nil?

# afficher du texte dans la console
puts "texte #{variable} " + variable.to_s

# quitter le script
exit

# test si objet possède la méthode methode
if objet.respond_to?("methode")

# Différence entre " " et ''
variable = 'V'
puts "#{variable} \n"    #=> V
puts '#{variable} \n'    #=> #{variable} \n
```

Boucle each

```ruby
tableau = ["A", "B", "C"]

tableau.each do |lettre|
  puts "#{lettre}"
end
```

Boucle for

```ruby
for i in 0..5
  puts i.to_s
end
```

Manipulation de chaînes de caractères

```ruby
maChaine = "un,deux.trois"
# remplacer les . et les , par des espaces
puts maChaine.gsub(/(,|\.)/, " ")    # => un deux trois
# récupérer une sous-chaîne allant de la position 4 jusqu'à la fin
sousChaine = maChaine[4..maChaine.length]
# string.contains()
if maChaine.include? 'deux'
```

http://ruby-doc.org/core/classes/String.html

Méthodes

```ruby
def methode(parametreDefaut = "defaut")
      puts "parametre : #{parametre}"
end

methode("parametre") # la méthode doit être définie avant son appel
methode "parametre"
methode()
```

Variables globales

```ruby
$variableGlobale = "VG"  # pour être globale, elle doit commencer par $

def methode()
      puts "#{$variableGlobale}"  # comme elle est globale, elle est accessible dans methode
end
```

Fichiers et dossiers

Lecture et écriture dans les fichiers

```ruby
# Affichage ligne par ligne du fichier
File.foreach('fichier.txt') do |line|
  puts line
end

# Stocker le contenu d'un fichier dans une chaîne de caractères
contenu = File.read('fichier.txt')
puts contenu

# Ecrire dans fichier.txt
fichier = File.open("fichier.txt", 'w')
fichier.write("texte")
```

http://ruby-doc.org/core/classes/File.html

http://ruby-doc.org/core/classes/IO.html

Fichiers

```ruby
File.basename('/dossier/fichier.txt')    # → fichier.txt

if File.exist?('fichier.txt')    # Tester si fichier.txt existe
File.delete 'fichier.txt' # Supprime fichier.txt
FileUtils.mv('fichier.txt', 'fichier ou dossier') # Déplace et/ou renomme
FileUtils.cp('fichier.txt', 'fichier ou dossier') # Copie
```

http://ruby-doc.org/core/classes/File.html

Dossiers

L'utilisation de FileUtils nécessite l'ajout de `require 'fileutils'`

```ruby
Dir.mkdir 'Nom du répertoire'    # Créer un dossier
Dir.pwd                          # Répertoire courant
Dir.chdir("Dossier")             # Changer de répertoire courant
if File.directory?('dossier')    # Tester si dossier existe et est bien un dossier
FileUtils.rm_rf 'dossier'        # Suppression d'un répertoire non-vide

# Copie récursive. Options : verbose, remove_destination
FileUtils.cp_r('source', 'destination', :verbose => true)

pathToAllFiles=File.join("Dossier", "*")   # Créer le chemin Dossier\*
files = Dir[pathToAllFiles] # Lister tous les fichiers de Dossier
lyricFiles = files.select { |file| /(.+\.lrc)$/ =~ file }
```

http://ruby-doc.org/core/classes/Dir.html

http://ruby-doc.org/core/classes/FileUtils.html

Classe Find

```ruby
Find.find('chemin') do |file|
  puts file    # Affiche tous les fichiers, dossiers de chemin, puis récursivement les sous-dossiers
end
```

Arguments du script

```ruby
# tableau contenant les arguments qui ont été passé au script
ARGV

# premier argument
ARGV[0]

# nombre d'arguments
ARGV.size
```

If..else, unless

```ruby
if var == 1     # les opérateurs suivant existent : not != && ||
      puts "1"
elsif var == 2
      puts "2"
else
      puts "3"
end

# code  if  condition
debug = 1
print "debug\n" if debug

unless var == 1     # équivalent à if not
      puts "1"
else
      puts "2"
end
```

Lancer un exécutable

method	stdout	stderr	stdin	real-time	comments	process id	exit code
exec("command")	no	no	no	yes	Simple, non-interactive invocation; replaces current process with new process (in other words, any Ruby code after the exec will not be executed!)	NA	NA
system("command")	no	no	no	yes	Simple, non-interactive invocation; waits till execution is done; outputs both stdout and stderr as normal	NA	$?.exitstatus
result = `command`	yes	no, unless you do 2>&1	no	Buffered - output is returned only when the command has finished/exited	Same, only it capture the output of that process.	NA	$?.exitstatus
pipe = IO.popen("command", "r")	yes	no, unless you do 2>&1	no	Yes - can even read a char or a line at a time, if you want	Interactive control of other process (write to its stdin, and then read from its stdout)	pipe.pid	$?.exitstatus
pipe = IO.popen("command", "w+")	yes	no, unless you do 2>&1	yes	""	""	""	""
Open3.popen3("command")	yes	yes	yes	yes	Very similar to IO.popen.		

```ruby
`exécutable`
`exécutable &> sortie.txt`

resultat = system 'exécutable'

sortie = %x[exécutable]
```

Fork

Sous Windows il faut préalablement installer la bibliothèque Win32Utils. (compatible ruby 1.8.x)

(gem install win32-process)

```ruby
require 'rubygems'
require 'win32/process'

`exécutable` if Process.fork.nil?
```

Hash (Dictionnaire)

```ruby
dictionnaire = { clé => valeur }
puts dictionnaire[clé]

dictionnaire.default = valeur   # définit une valeur par défaut pour les clés inéxistantes

if dictionnaire.key?(clé)   # test si le dictionnaire contient la clé
```

Expressions rationnelles

```ruby
regex = Regexp.new('expression rationnelle')
if regex.match("string")
if "string".match(regex)
if "string".match =~ /expression rationnelle/
if /expression rationnelle/.match("string")
#  options : i → non sensible à la casse, m, x, o
if /expression rationnelle/i =~ "string"

# Récupérer l'expression rationnelle sous la forme d'une chaîne de caractères
regex.source.to_s

# Afficher le résultat du match
puts $&
```

Aide mémoire

`. () [] {} \ ?*`	Caractères spéciaux sont à préfixer d'un \
`[abc]`	Au moins un caractère parmi a, b ou c
`[^abc]`	Au moins un caractère qui n'est pas a, b ou c
`[a-z]`	Au moins un caractère dans la liste allant de a à z
`[a-zA-Z]`	Au moins un caractère dans la liste allant de a à z ou dans la liste allant de A à Z
`.`	Joker
`\s`	Espace. `\S` Tous les caractères à l'exception de l'espace.
`\w`	Lettre, nombre ou _. `\W` Tous les caractères sauf lettre, nombre ou _
`(x)$`	Se terminant par x
`^(x)`	Commençant par x
`(a|b)`	a ou b
`x?`	0 ou 1 fois x
`x*`	0 ou plusieurs fois x. `x*?` version non-gloutonne.

`x+`	Au moins 1 fois x. `x+?` version non-gloutonne.
`x{6}`	6 fois x
`x{6,}`	Au moins 6 fois x
`x{3,6}`	Entre 3 et 6 fois x
`(?=...)`	Assertion look-ahead, correspond si la position actuelle dans la chaîne de caractères est suivie par ... Exemple: "Isaac(?=Asimov)" correspondra à "Isaac" seulement si il suivi de "Asimov". Pas d'assertion look-behind en ruby.
`(?!...)`	Assertion look-ahead négative, correspond si la position actuelle dans la chaîne de caractères n'est pas suivie par ...

http://www.zenspider.com/Languages/Ruby/QuickRef.html#11

http://ruby-doc.org/core/classes/Regexp.html

http://ruby-doc.org/core/classes/MatchData.html

Test en ligne : http://www.rubyxp.com/

Classes

```ruby
class MaClasse
	attr_accessor :texte	# permet d'acceder à la variable d'instance texte

	def initialize(texte = "...")	# constructeur de l'objet
		@texte = texte
	end

	def affiche
		puts "#{@texte}"
	end
end

# utilisation de la classe
maClasse = MaClasse.new	# constructeur sans paramètres
maClasse = MaClasse.new "MonTexte"

maClasse.texte = "Nouveau texte"
puts "attribut texte : #{maClasse.texte}"

maClasse.affiche
```

Ftp

```ruby
require 'net/ftp'

Net::FTP.open('adresse ftp', 'identifiant', 'mot de passe') do |ftp|
   ftp.debug_mode=true # Log dans la console la communication avec le serveur ftp
   ftp.passive=true
   ftp.chdir('dossier') # Se rendre dans le répertoire dossier
   ftp.putbinaryfile('fichier local', 'nom du fichier distant')
   ftp.close
end

ftp = Net::FTP.new'adresse ftp', 'identifiant', 'mot de passe')
ftp.puttextfile('fichier local', 'nom du fichier distant')
ftp.close
```

http://ruby-doc.org/stdlib/libdoc/net/ftp/rdoc/index.html

Tk

Permet d'afficher des interfaces graphiques. Installer *libtcltk-ruby*.

http://www.tutorialspoint.com/ruby/ruby_tk_guide.htm

```ruby
require 'tk'

root = TkRoot.new { title "Titre" }
TkLabel.new(root) do
   text 'Message'
   pack { padx 150 ; pady 15; side 'left' }
end
Tk.mainloop
```

Ruby Console vs Ruby Window

Il existe 2 interpréteurs ruby :

- ruby.exe affiche la console
- rubyw.exe lance un script ruby sans la console

Menu : HighLine

Installer HighLine : `gem install highline`

```ruby
require 'rubygems'
require 'highline/import'

choose do |menu|
      menu.prompt = "Votre choix > "
      menu.choice(:'Bonjour') { say("Bonjour!") }
      menu.choice(:'Quitter') { say("Au revoir!") }
end
```

REXML

Outils de parsing XML inclut dans l'installation de Ruby depuis la version 1.8.

```ruby
require 'rexml/document'
include REXML  # équivalent du using C#

file = File.open('fichier.xml')
doc = Document.new(file) # lecture et parsing du XML
file.close()

puts XPath.first(doc, '/Noeud1/Noeud2/text()' ) # requête XPath
doc.root.elements['Noeud2'].text = "NouveauTexte" # modification du XML

formatter = REXML::Formatters::Default.new
File.open('fichier.xml', 'w') do |newFile|
      formatter.write(doc, newFile) # écriture de l'XML modifié
end
```

http://www.germane-software.com/software/XML/rexml/doc/

Script Nautilus

La sortie standard est redirigée vers *~/.xsession-errors*

```ruby
#!/usr/bin/env ruby
```

```
require 'tk'

ENV['NAUTILUS_SCRIPT_SELECTED_FILE_PATHS'].split("\n").each do |path|
      ...
end
```

Installation

Ruby installer

Installation des exécutables de base pour Ruby.

Version 1.9.1 n'est pas compatible avec la Gem win32-process.

Version 1.8.7 OK.

One-click installer

Seulement disponible pour la version 1.8.6

Binaire

Déconseillé. Installer RubyGems : télécharger et dé-zipper l'archive. Puis exécuter `ruby setup.rb`

GEM

```
gem install paquet
gem uninstall paquet
gem install -l fichier.gem  # installation d'un fichier gem
gem list                    # lister les gem installées
gem update                  # mettre à jour toutes les gems
```

IronRuby

Projet permettant d'utiliser le Framework .NET dans les script Ruby, mais aussi d'intégrer des script Ruby dans le code .NET.

Pour utiliser une version de ruby préalablement installée, modifier les chemins de la balise <options><set> dans le fichier « ir.exe.config ».

WinForm

```
require 'mscorlib'
require 'System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
         PublicKeyToken=b77a5c561934e089'
require 'System.Drawing, Version=2.0.0.0, Culture=neutral,
         PublicKeyToken=b03f5f7f11d50a3a'
# le require peut aussi s'écrire
require
'C:\Windows\Microsoft.NET\Framework\v2.0.50727\System.Windows.Forms.dll'

Application = System::Windows::Forms::Application
Form = System::Windows::Forms::Form
MessageBox = System::Windows::Forms::MessageBox
Button = System::Windows::Forms::Button
Point = System::Drawing::Point

class MyForm < Form
  def initialize
    self.text = "My .NET Form from Ruby"

    @button = Button.new
    @button.location = Point.new 150, 150
    @button.text = "Click Me!"

    my_click_handler = Proc.new {|sender, e| MessageBox.show 'Hello from
Ruby!'}
    @button.click(&my_click_handler)

    self.controls.add @button
  end
end

my_form = MyForm.new
Application.run my_form
```

Regex

```
require 'C:\Windows\Microsoft.NET\Framework\v2.0.50727\System.dll'
Regex = System::Text::RegularExpressions::Regex

# définition de l'expression rationnelle
regex = Regex.new('(?<=(?<!// )\[assembly: AssemblyVersion\(\")([0-9]+\.)
{2,3}([0-9]+|\*)(?=\"\)\])')

# recherche de l'expression dans le texte
match = regex.Match(texte_a_tester)

match.Success # test si l'expression a été trouvé
match.Value # la valeur de l'expression trouvée

# remplace l'expression trouvée
texte_remplacé = regex.Replace(texte_original, texte_de_remplacement)
```

Netbeans

Il existe un pack Netbeans pour Ruby.

Il est possible d'utiliser JRuby ou bien Ruby.

Install Fast Debugger

Impossible de l'installer sur Windows!!!

Faire l'installation à la main. Télécharger « ruby-debug-base-0.10.3.1-java.gem » puis l'installer :

```
jruby -S gem install -l ruby-debug-base-0.10.3.1-java.gem
```

Installer ruby-debug-ide

```
jruby -S gem install --ignore-dependencies -v 0.4.6 ruby-debug-ide
```

Source : http://wiki.netbeans.org/RubyDebugging#Quickstart.2FInstallation

Bug : Don't know how to build task '2>&1'

```
Building native extensions.  This could take a while...
rake aborted!
Don't know how to build task '2>&1'
```

Problème rencontré avec jruby 1.4

Correction : `set rake=jruby -S rake`

Bug : 'cl' n'est pas reconnu en tant que commande

Ajouter au path :

- c:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE;
- c:\Program Files\Microsoft Visual Studio 9.0\VC\BIN;
- c:\Program Files\Microsoft Visual Studio 9.0\Common7\Tools;
- c:\Windows\Microsoft.NET\Framework\v3.5;
- c:\Windows\Microsoft.NET\Framework\v2.0.50727;
- c:\Program Files\Microsoft Visual Studio 9.0\VC\VCPackages;

Source : C:\Program Files\Microsoft Visual Studio 9.0\Common7\Tools\vsvars32.bat

Autres IDE

- RubyMine (jetbrain IntellJ, payant, Windows et Linux)
- Ruby In Steel (Visual Studio, payant)
- Aptana (impossible de créer un projet ruby)
- Eclipse (impossible d'installer plugin Ruby Development Tool)
- Ruby Developement Environment (2007, gratuit, pas d'auto-complétion)