

# IRB (Interactive Ruby Shell)

Interactive Ruby ou irb est un interpréteur de commandes qui permet d'exécuter des commandes Ruby de manière interactive. Il peut être utilisé pour tester le langage. Il est fourni avec la version officielle de Ruby. Une version existe également sur le Web.

IRB permet d'écrire et d'interpréter du Ruby à la volée, sans avoir à enregistrer un fichier et l'exécuter ensuite.

# Usage

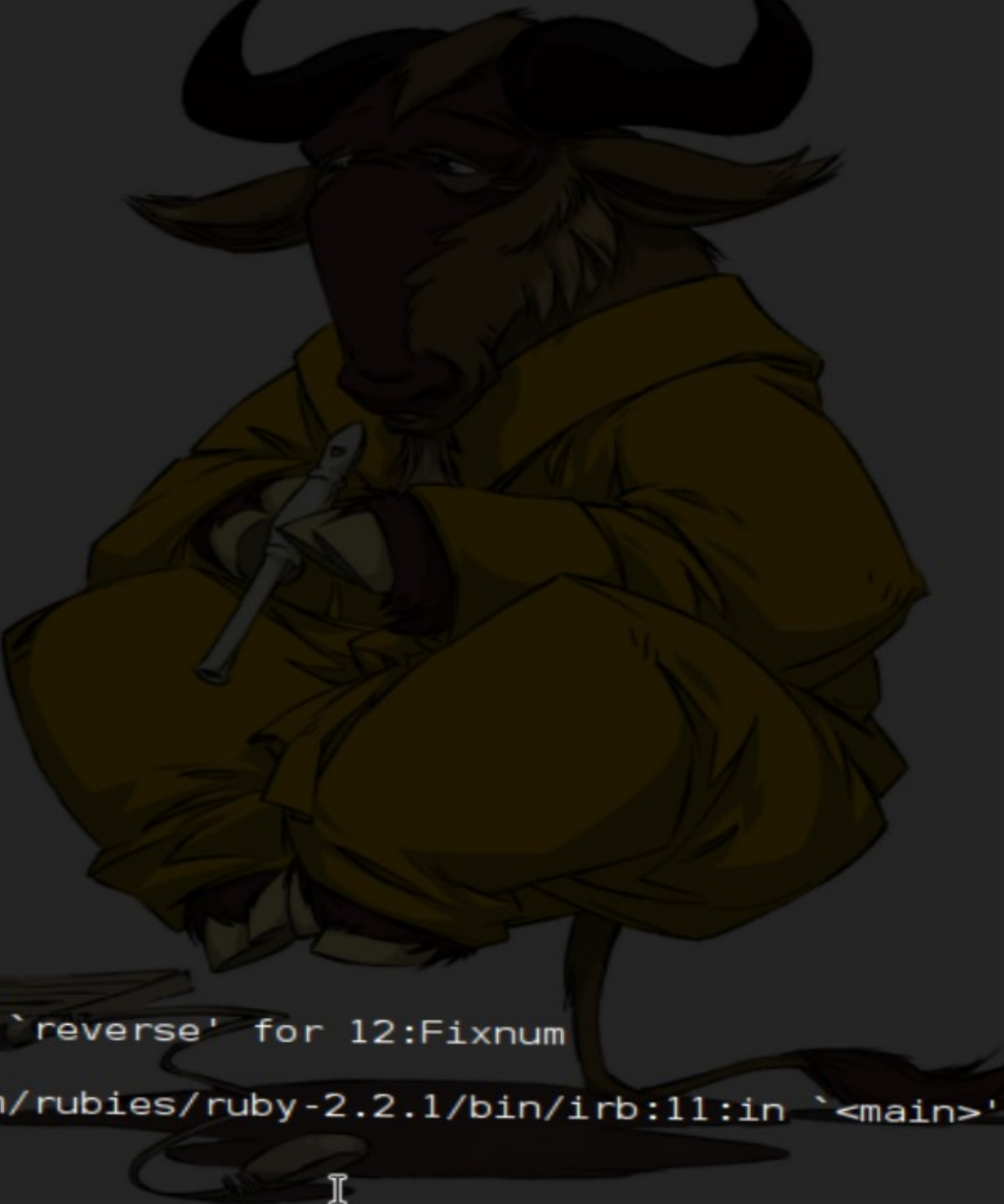
Lors de l'exécution de IRB, les invites sont affichées comme suit (diapo suivante).

Ensuite, entrez l'expression Ruby. Une entrée est exécutée quand la syntaxe est complète.

Pour quitter IRB, il suffit d'entrer exit.

## Terminal

Fichier Éditer Affichage Terminal Onglets Aide



```
handyrod@handylinux ~  
$ irb  
2.2.1 :001 > 1+2  
=> 3  
2.2.1 :002 > class Foo  
2.2.1 :003?>   def foo  
2.2.1 :004?>     print 1  
2.2.1 :005?>     end  
2.2.1 :006?>   end  
=> :foo  
2.2.1 :007 > nil  
=> nil  
2.2.1 :008 > 2 * 3  
=> 6  
2.2.1 :009 > "rodolphe"  
=> "rodolphe"  
2.2.1 :010 > name = "rodolphe"  
=> "rodolphe"  
2.2.1 :011 > name.capitalize  
=> "Rodolphe"  
2.2.1 :012 > name  
=> "rodolphe"  
2.2.1 :013 > name.reverse  
=> "ehplodor"  
2.2.1 :014 > name * 3  
=> "rodolpherodolpherodolphe"  
2.2.1 :015 > 12.reverse  
NoMethodError: undefined method `reverse' for 12:Fixnum  
    from (irb):15  
    from /home/handyrod/.rvm/rubies/ruby-2.2.1/bin/irb:11:in `'  
2.2.1 :016 > 12.to_s.reverse  
=> "21"  
2.2.1 :017 > 
```

Il est très facile d'utiliser IRB comme une vulgaire calculatrice

```
irb(main):003:0> 3+2
```

```
=> 5
```

```
irb(main):004:0> 3*2
```

```
=> 6
```

Symbole	Effet
+	addition
-	soustraction
*	multiplication
/	division
**	à la puissance de (exposant
%	modulo (le reste d'une division)

```
irb(main):006:0> Math.sqrt(9)
```

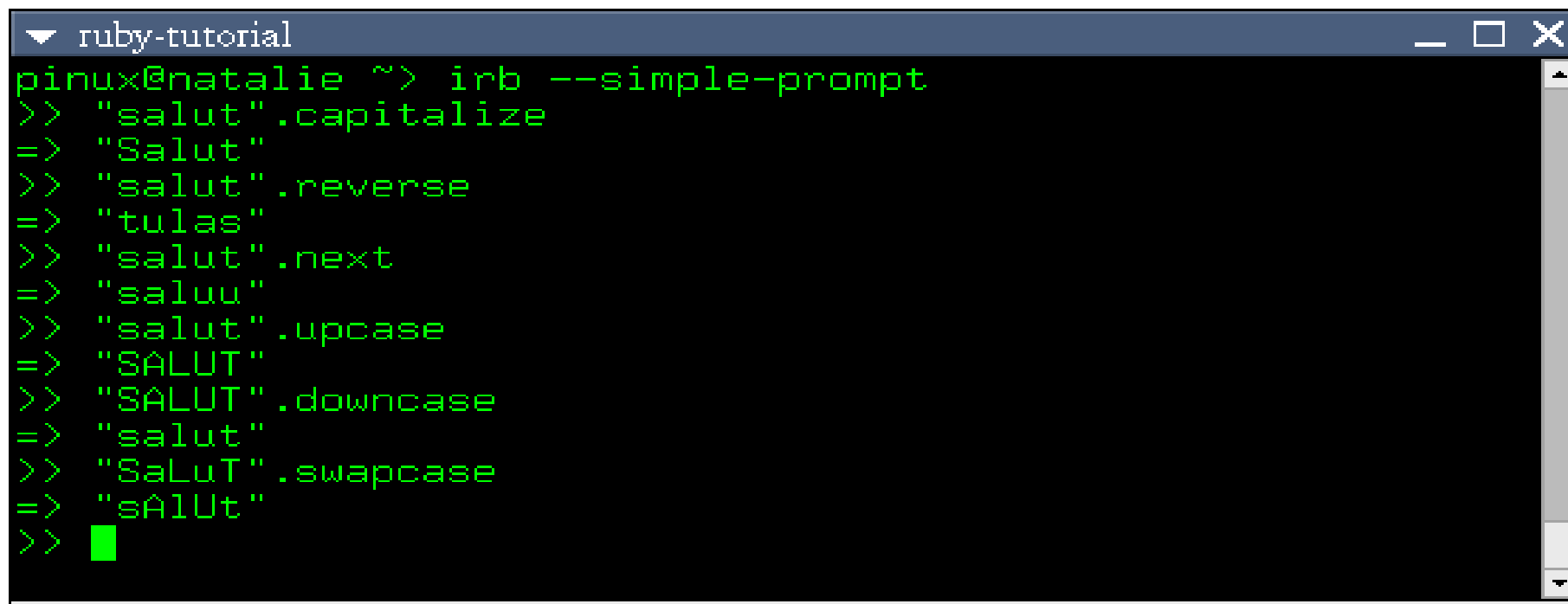
```
=> 3.0
```

(Math.sqrt() pour la racine carré)

Math est ce qu'on appelle en Ruby, un module.

# Chaîne de caractères

Les mots ou blocs de texte sont identifier par le terme *chaîne de caractère* (« *String* »)

A screenshot of a terminal window titled 'ruby-tutorial'. The window shows a series of Ruby commands and their outputs. The commands are: 'irb --simple-prompt', '"salut".capitalize', '"salut".reverse', '"salut".next', '"salut".upcase', '"SALUT".downcase', and '"SaLuT".swapcase'. The outputs are: "Salut", "tulas", "saluu", "SALUT", "salut", and "sAlUt". The terminal has a dark background with green text. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

```
▼ ruby-tutorial
pinux@natalie ~> irb --simple-prompt
>> "salut".capitalize
=> "Salut"
>> "salut".reverse
=> "tulas"
>> "salut".next
=> "saluu"
>> "salut".upcase
=> "SALUT"
>> "SALUT".downcase
=> "salut"
>> "SaLuT".swapcase
=> "sAlUt"
>> █
```

# Types de données et leur classes

Type de données	Classe associée
Nombre entier	Integer
Nombre décimal	Float
texte	String

# Classes et quelques méthodes

Classe	Quelques méthodes
Integer	+, -, /, *, %, **
Float	+, -, /, *, %, **
String	capitalize, reverse, length, upcase, +, *

# Méthodes de conversions entre classes

Méthode	Conversion		
	D'un(e)	Vers un(e)	
String#to_i	Chaîne de caractère	Entier	
String#to_f	Chaîne de caractère	Flottant	
Float#to_i	Flottant	Entier	
Float#to_s	Flottant	Chaîne de caractère	
Integer#to_f	Entier	Flottant	
Integer#to_s	Entier	Chaîne de caractère	



# Les variables

Ruby vous permet d'utiliser des variables pour associer des noms à des objets particuliers.

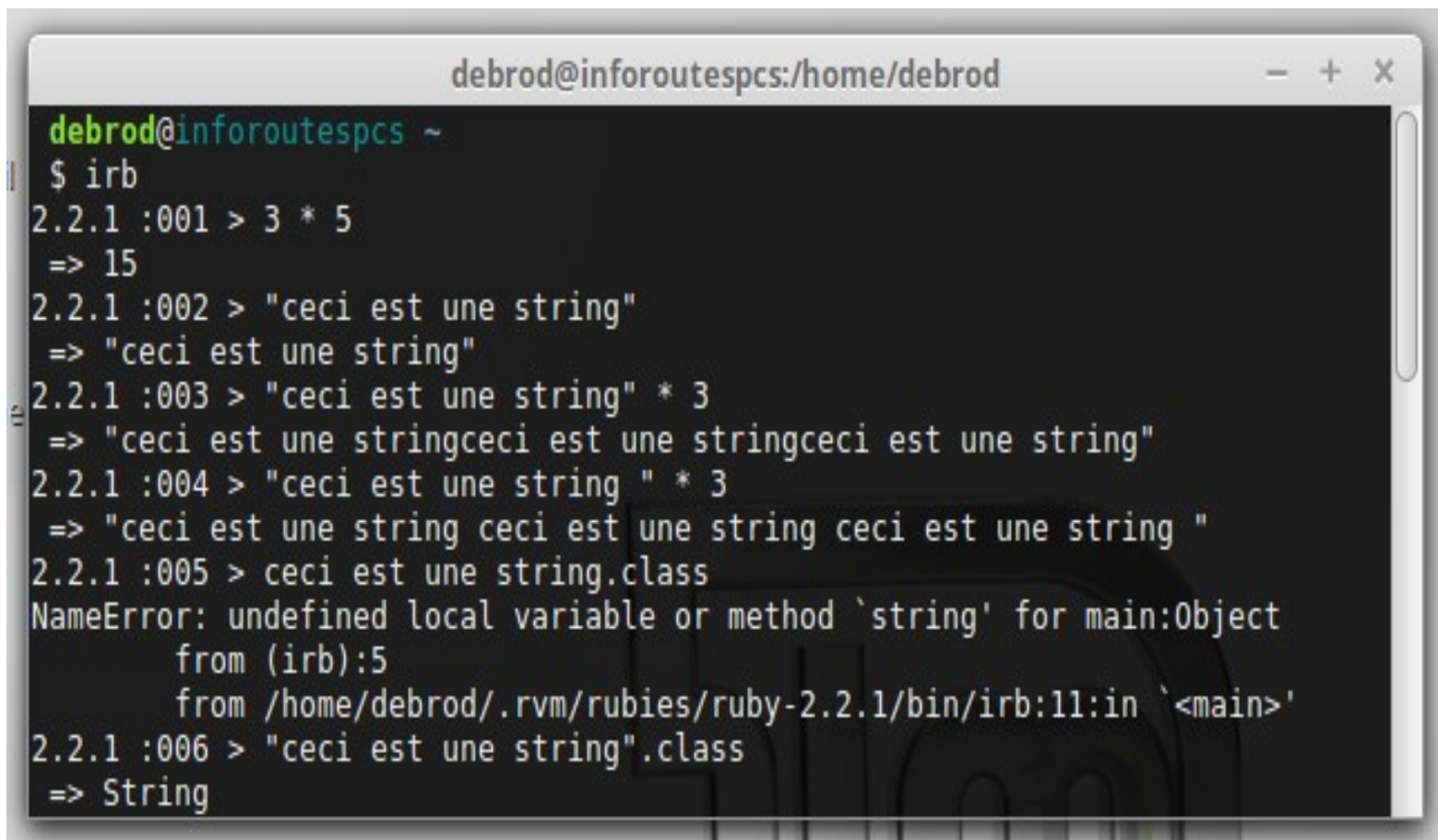
Exemple :

`nom = « Toto el Montoso »`

Imaginez-vous Ruby fabriquant deux tableaux. Un contenant les objets, et un autre contenant les noms qui leurs sont associés. Ensuite, imaginez-vous Ruby dessinant une flèche de ville à "Toronto".

Attention, les noms de variables doivent toujours commencer par une minuscule !

Les commandes que nous avons explorées ensemble.  
Pour les Strings.

A screenshot of a terminal window with a title bar that reads 'debrod@inforoutespcs:/home/debrod'. The terminal shows a series of commands entered in the Ruby Interactive (IRB) shell. The first command is '\$ irb', which starts the IRB session. Subsequent commands include arithmetic operations like '3 \* 5' and string operations like concatenation and multiplication. One command, 'ceci est une string.class', results in a 'NameError' because 'string' is not a defined variable or method in that context. The final command, '"ceci est une string".class', correctly returns the 'String' class.

```
debrod@inforoutespcs ~  
$ irb  
2.2.1 :001 > 3 * 5  
=> 15  
2.2.1 :002 > "ceci est une string"  
=> "ceci est une string"  
2.2.1 :003 > "ceci est une string" * 3  
=> "ceci est une stringceci est une stringceci est une string"  
2.2.1 :004 > "ceci est une string " * 3  
=> "ceci est une string ceci est une string ceci est une string "  
2.2.1 :005 > ceci est une string.class  
NameError: undefined local variable or method `string' for main:Object  
    from (irb):5  
    from /home/debrod/.rvm/rubies/ruby-2.2.1/bin/irb:11:in `'  
2.2.1 :006 > "ceci est une string".class  
=> String
```

Ici aussi les commandes que nous avons essayées, avec des Arrays (des listes, tableaux) et des commentaires avec le # (qui commente), mais nous permet de ne plus avoir de retour nil...

```
debrod@inforoutespcs:/home/debrod
$ irb
2.2.1 :001 > 3.to_s + "3"
=> "33"
2.2.1 :002 > [2, 4, 5]
=> [2, 4, 5]
2.2.1 :003 > [2, 4, 5].class
=> Array
2.2.1 :004 > [2, 4, 5].count
=> 3
2.2.1 :005 > 2
=> 2
2.2.1 :006 > puts 2
2
=> nil
2.2.1 :007 > puts 2 ; puts 2 ; puts 3 ; "foo"
2
2
3
=> "foo"
2.2.1 :008 > array = [2, 3, 9]
=> [2, 3, 9]
2.2.1 :009 > array.class
=> Array
2.2.1 :010 > a = ["a", "b", "c"]
=> ["a", "b", "c"]
2.2.1 :011 > a.slice!(1)      #=> "b"
=> "b"
2.2.1 :012 > a               #=> ["a", "c"]
=> ["a", "c"]
2.2.1 :013 > a.slice!(-1)    #=> "c"
=> "c"
2.2.1 :014 > a.slice!(100)   #=> nil
=> nil
2.2.1 :015 > a              #=> ["a"]
=> ["a"]
2.2.1 :016 >
```

# Conclusion

IRB, est indispensable je pense pour commencer à apprendre le langage de Ruby.

Il permet de tester des méthodes lorsque on écrit un programme pour vérifier les résultats et les fonctionnements du code.

# Liens pour découvrir et aller plus loin.....

- <http://ruby-doc.org/docs/beginner-fr/xhtml/pr01.html>
- <https://www.ruby-lang.org/fr/documentation/quickstart/>
- <http://rubykoans.com/>
- [http://www.tutorialspoint.com/ruby/interactive\\_ruby.htm](http://www.tutorialspoint.com/ruby/interactive_ruby.htm)
- <http://ruby-doc.org/stdlib-2.2.2/libdoc/irb/rdoc/index.html>
-