Atelier du libre
Programmer avec Ruby

Paul Rivier
paul.r.ml@gmail.com

Mai 2009

Plan

Informatique

Étymologie

Informatique

De la contraction des mots ***« information »*** *et* ***« automatique »****. Inventé par Philippe Dreyfus en 1962, il fut officiellement consacré par Charles de Gaulle qui trancha lors d'un Conseil des ministres entre « informatique » et « ordinatique ».*

Définition

Informatique, nom commun, féminin

Domaine des concepts et autres techniques employées pour le traitement automatique de l'information.

Motivations

calcul	répéter rapidement un traitement de données
réseaux	communiquer à distance
données	expression relationnelle, duplication, recherche
...	*beaucoup d'autres choses*

Histoire de l'informatique

Histoire ancienne : systèmes mécaniques

Avant XVI^{e}	bouliers, régles de calculs
XVI^{e}- XVII^{e}	premières calculettes mécaniques addition/soustration (type « pascaline »)
XVIII^{e}- XX^{e}	calculettes élaborées, constructions de tables (logarithme, trigonométrie)
XVIII^{e}	première machine programmable, un métier à tisser qui lit un ruban perforé

Histoire moderne : systèmes électroniques

1930 - 1956 calculateur électroniques à lampes

1956 - 1971 calculateurs électroniques programmables à transistors

1970 - aujourd'hui micro-informatique

Histoire moderne : les langages

1950 - 1970 Fortran, LISP, ALGOL, COBOL, Simula, BASIC

1970 - 1980 C, Smalltalk, Prolog, ML

1980 - 2000 C++, Eiffel, Haskell, Ruby, ANSI CL ...

Depuis la micro-informatique, le support materiel a peu évolué, et le travail sur le techniques de programmation logicielle a acceléré.

Les langages

Les usages des langages

système [noyaux, pilotes] Priorité au contrôle du materiel et à la vitesse (C ou C++)

calcul [algorithmes, simulations] Priorité à la vitesse de calcul et à la gestion de la mémoire

applications [navigateur, mail, dessin, ...] Priorité à l'expressivité et à l'abstraction (Ruby)

Ruby est un langage open-source dynamique qui met l'accent sur la simplicité et la productivité.

Concepts élémentaires

Le fichier source

- un « langage » est un programme dont l'entrée est un fichier source
- on peut utiliser plusieurs fichiers sources en les référençant

Mon premier programme

- ouvrir un éditeur de texte
- taper `puts "J'aime les ateliers du libre"`
- enregistrer dans `chemin/vers/fichier.rb`
- ouvrir une console et taper `ruby chemin/vers/fichier.rb`
- insérer en première ligne `# !/usr/bin/ruby -w`, puis faites `chmod +x fichier.rb` pour pouvoir faire `./fichier.rb`

Avant d'aller plus loin

Le développement se fait avec un gestionnaire de sources. Ce dernier va vous permettre de :

- programmer de façon incrémentale
- revenir en arrière en cas de problème
- maintenir plusieurs axes de développement concurrents
- assembler ces axes si besoin
- publier votre code
- et bien plus ...

Mercurial

Allez dans votre dossier `chemin/vers/` puis tapez

- `hg init`
- `hg add fichier.rb`
- `hg ci -m "mon premier programme"`
- `hg view`

L'instruction

- une instruction est un mot reconnu par le langage
- en ruby, les instructions principales sont `alias`, `and`, `begin`, `break`, `case`, `class`, `def`, `do`, `else`, `elsif`, `end`, `false`, `for`, `if`, `in`, `module`, `next`, `nil`, `not`, `or`, `return`, `self`, `super`, `then`, `true`, `unless`, `until`, `when`, `while`, `yield`

Entrées et sorties

entrée le programme acquiert de l'information depuis l'extérieur

sortie l'extérieur acquiert de l'information depuis le programme

Entrées et sorties (garder #!...)

```
puts "ecris ton prenom et tape entree"
prenom = gets.chomp
puts "ton prenom est #{prenom}"
```

Unités logiques et nommage

variable conteneur de données

fonction conteneur d'instructions paramétrable

Variables et fonctions (tout virer)

```
def bonjour(texte)
  return "Bonjour, " + texte
end

ma_var = "J'aime les ateliers du libre."
puts bonjour(ma_var)
```

Types

- chaque variable a un type
- chaque type a sa collection de méthodes
- les comparaisons doivent utiliser des types compatibles
- types de base : entier, nombre décimal (approché), chaîne de caractères, booléen (vrai ou faux), tableau ...

Types (à essayer dans irb)

```
5.class
"bonjour".class
5>4
5>"bonjour"
[2, 4, 6]
```

Algorithmique

- controle d'exécution
- branchements conditionnels
- boucles

Boucles et branchements (tout garder)

```
puts "nombre de repetitions ?"
n = gets.chomp.to_i
if n>5 then
  puts "c'est trop"
  n=5
end
n.times do
  puts bonjour(ma_var)
end
```

Les tableaux

- un tableau est une collection ordonnée
- en ruby, un tableau peut contenir tous types d'éléments, mélangés
- y compris d'autres tableaux

Les tableaux

Les tableaux (tout virer sauf def bonjour)

```
continuer=true
tableau=Array.new
while continuer
  t=gets.chomp
  if t!=""
    tableau.push t
  else
    continuer=false
  end
end
tableau.each { |t| puts bonjour(t) }
```

Chaînes de caractères

- utiles pour récupérer une entrée ou préparer une sortie

Chaînes de caractères (dans irb)

```
a="Bonjour"
a.size
a.upcase
b="salut les amis"
b.split(" ")
b.gsub("amis","libristes")
```

Structures associatives

- structures conçues pour associer une valeur à une clef
- le tableau associatif est le plus simple
- la table de hashage est optimisée pour les grandes collections

Structures associatives (dans irb)

```
h=Hash.new
h["paul"]="Paul parle des structures associative
h["Christophe"]="Christophe regarde rigolant"
h
puts h["Paul"]
puts h["Christophe"]
```

On continue sans vignettes