**G. Hunault**
Master 1 Informatique

*Angers, avril 2015*

# Langages de scripts et Frameworks de développement Web

## 1. Programmation RePyRu

Dans le but de comparer les langages REXX/REGINA, PYTHON et RUBY on se propose ici d'écrire la même fonction dans ces 3 langages. On suppose qu'on dispose au départ de la fonction **cip** écrite en PHP dont voici le code-source (cip est mis pour **constraint integer programming**) :

```
function cip($n=20,$vmin=1,$vmax=100) {

     $tabN = array() ;
     for ($idn=1;$idn<=$n;$idn++) {
       $tabN[$idn-1] = rand($vmin,$vmax) ;
     } # fin pour idn
     return($tabN) ;

} # fin de fonction cip
```

Cette fonction renvoie donc un tableau de n nombres **entiers** entre vmin et vmax obtenus par un tirage pseudo-aléatoire, avec par défaut 20 nombres entiers entre 1 et 100.

a) Ecrire une fonction REXX/REGINA de même nom qui réalise la même tâche ; exemple d'utilisation : tnbA = cip(20,1,100).

b) Ecrire une fonction PYTHON 3+ de même nom qui réalise la même tâche ; exemple d'utilisation : tnbB = cip(20,1,100).

c) Ecrire une **méthode** RUBY 1.9+ de même nom dans la classe *Fixnum* qui réalise la même tâche ; exemple d'utilisation : tnbC = 20.cip(1,100).

Pour Python et Ruby, on utilisera de préférence map.

## 2. Un peu d'ORM en Ruby pour Rails

On suppose qu'il existe une table elves dans la base de données MySQL nommée scripts. Indiquer par quoi il faut remplacer les caractères XXXXX dans le script suivant pour obtenir l'affichage demandé.

```
require 'active_record'

ActiveRecord::Base.establish_connection(
   :adapter  => "mysql",
   :host     => "localhost",
   :database => "scripts",
   :username => "anonymous",
   :password => "anonymous"
) # fin de ActiveRecord

class Elf < ActiveRecord::Base
  # pour profiter de l'ORM
end # fin de classe Elf

# ------------------------------------------------

puts " les trois premiers ID et AGE des femmes (SEXE=1) de plus de " ;
puts " soixante ans par ordre d'ID "

## on veut reproduire ce qu'afficherait la requete
## SELECT iden, age FROM elf
##        WHERE sexe=1 AND age>60 ORDER BY iden LIMIT 3

Elf.XXXXX.each { |enr|

  puts enr.IDEN + " " + enr.AGE.to_s

} # fin each
```

On viendra bien sûr chainer les méthodes objets nommées *where()*, *order()* et *limit()*.

## 3. Des tests pour **Ruby on Rails**

Dans le cadre d'une application développée en Ruby on Rails, on rapatrie automatiquement des séquences FASTA à partir de sites américains. Les séquences intéressantes pour l'application sont des séquences d'ADN (des chaines de caractères qui ne contiennent que les lettres A, C, G ou T), "pas trop petites" (plus de 50 caractères) et "pas trop grandes" (moins de 400 caractères).

Indiquez quel test ou quels tests vous pouvez écrire en Ruby pour vérifier que les imports fournissent bien les séquences escomptées. On pourra supposer que l'application a été créée par

```
rails new genomicApp
```

Vous indiquerez où il faut écrire ces tests et comment on les exécute.

## 4. Discussion sur la notion de **framework**

Les frameworks récents bâtis sur Python et Ruby comme Django et Rails sont souvent nommés frameworks de développement du coté serveur. Deux questions viennent naturellement à l'esprit :

*Peut-on s'en servir comme frameworks de production ?*

*Que penser des frameworks coté client ?*

Rédigez une réponse à chacune de ces deux questions, si possible à l'aide d'exemples concrets. Il est conseillé d'utiliser au moins trois mots de trois syllabes ou plus pour « transmettre un contenu rédactionnel fort ». Une dizaine de lignes parait être une rédaction minimale.