



Initiation à la programmation en Ruby - 42

Jour 01

Staff 42 bocal@42.fr

Résumé: Ce document est le sujet du jour 01 de la piscine d'initiation à la programmation en Ruby.

Table des matières

| | | |
|-------------|---|----------|
| I | Consignes | 2 |
| II | Exercice 00 : La base | 3 |
| III | Exercice 01 : mon premier script | 4 |
| IV | Exercice 02 : name | 5 |
| V | Exercice 03 : name++ | 6 |
| VI | Exercice 04 : what's your name | 7 |
| VII | Exercice 05 : UPCASE_IT | 8 |
| VIII | Exercice 06 : age | 9 |


Chapitre I

Consignes

- A 42, vous allez faire l'expérience d'une pédagogie un peu particulière : vous avez un "cours" d'introduction d'1h tous les matins, et le reste de la journée, vous avez des exercices à réaliser en autonomie.
- Vous avez une question ? Un problème ? Un blocage ? Demandez à votre voisine de droite. Sinon, essayez avec votre voisine de gauche. A 42, les étudiants ne sont pas en compétition, ils avancent ensemble, en s'entre-aidant.
- Votre manuel de référence s'appelle **Google / man / Internet /** Vous allez devoir apprendre à faire des recherches sur internet, toutes les infos dont vous avez besoin s'y trouvent !
- Lisez attentivement les exemples. Vous devez respecter le formatage des réponses : les majuscules, les retours à la ligne... tout est important. Programmer, c'est avant tout faire preuve de rigueur.
- Un tuteur vous accompagne tout au long de la piscine : il/elle est là pour vous donner des pistes, vous indiquer comment faire vos recherches sur internet et vous encourager si vous êtes démotivées. Le tuteur est un soutien pour vous, mais il n'est pas là pour vous donner les réponses !
- A la fin de la journée, le tuteur de votre rangée va corriger votre travail collectivement. C'est un bon moment pour échanger et s'expliquer, entre élèves, les erreurs que vous avez pu faire, ou au contraire expliquer aux autres ce que vous avez compris. Soyez attentives.
- Bon courage, et n'ayez pas peur de vous tromper ! Faites des tests, tatonnez - en informatique, c'est en faisant des erreurs qu'on apprend !

Chapitre II

Exercice 00 : La base

| | |
|---|---------------|
|  | Exercice : 00 |
| On (re)prend de bonnes habitudes | |
| Dossier de rendu : <i>ex00/</i> | |
| Fichiers à rendre : aucun | |
| Fonctions Autorisées : Aucune | |
| Remarques : n/a | |


- Retournez dans le dossier **piscine_octobre** que vous avez créé hier.
- Créez un nouveau dossier **jour01** et déplacez vous dedans.
- Créez un nouveau dossier **ex00**. Ne mettez rien dedans.
- A partir de maintenant, tous les exercices suivants devront être dans le bon dossier de rendu. L'exercice 01 dans le dossier **ex01**, l'exercice 02 dans le dossier **ex02**, etc... vous avez compris la logique.



Vous devez réaliser cet exercice avec `mkdir` et `cd`, sinon ça n'a aucun intérêt :)

Chapitre III

Exercice 01 : mon premier script

| | |
|---|---------------|
|  | Exercice : 01 |
| Afficher 42 en Ruby | |
| Dossier de rendu : <i>ex01/</i> | |
| Fichiers à rendre : 42.rb | |
| Fonctions Autorisées : Toutes | |
| Remarques : n/a | |

- Créez un script **42.rb**.
- Ce script doit être exécutable. (pensez aux permissions notamment)
- Lorsqu'on l'exécute, le script doit afficher "42" suivi d'un retour à la ligne.

```
?> ./42.rb | cat -e
42$
?>
```




Ca vous paraît simple? C'est normal. Le ruby est un langage très proche de l'anglais, cela rend les recherches plus faciles.



Vous avez une petite ligne à ajouter au tout début de votre fichier pour que le shell sache comment interpréter le contenu du script. Cherchez sur Google "Making a Ruby Script Executable".

Chapitre IV

Exercice 02 : name


| | |
|---|---------------|
|  | Exercice : 02 |
| Afficher un nom | |
| Dossier de rendu : <i>ex02/</i> | |
| Fichiers à rendre : name.rb | |
| Fonctions Autorisées : Toutes | |
| Remarques : n/a | |

- Créez un script `name.rb`.
- Définissez une variable appelée `first_name`, initialisée avec votre prénom.
- Définissez une autre variable appelée `last_name`, initialisée avec votre nom.
- Affichez les variables, suivies d'un retour à la ligne.

```
?> ./name.rb | cat -e
Frederic Grati$
?>
```

Chapitre V

Exercice 03 : name++

| | |
|---|---------------|
|  | Exercice : 03 |
| Afficher un nom, en mieux | |
| Dossier de rendu : <i>ex03/</i> | |
| Fichiers à rendre : name.rb | |
| Fonctions Autorisées : Toutes | |
| Remarques : n/a | |

- Vous allez modifier votre script `name.rb`. (Après l'avoir copié dans le bon répertoire de rendu, bien sur !)
- Votre script va concaténer votre nom et votre prénom et affecter le résultat à une troisième variable appelée `whole_name`.
- Vous afficherez ensuite la variable `whole_name`, suivie d'un retour à la ligne.

```
?> ./name.rb | cat -e  
Frederic Grati$  
?>
```




Faites un man cp.



Cherchez sur internet comment concaténer des strings en Ruby.

Chapitre VI

Exercice 04 : what's your name

| | |
|---|---------------|
|  | Exercice : 04 |
| Comment t'appelles-tu ? | |
| Dossier de rendu : <i>ex04/</i> | |
| Fichiers à rendre : <code>whatsyourname.rb</code> | |
| Fonctions Autorisées : Toutes | |
| Remarques : n/a | |

- Créez le script `whatsyourname.rb`
- Ce script va demander d'abord à l'utilisateur d'entrer son nom, puis son prénom, et enfin affiche les deux.


```
?> ./whatsyourname.rb
Hey, what's your first name ? : Laurie
And your last name ? : Mezard
Well, pleased to meet you Laurie Mezard.
?>
```



Google gets, chomp.

Chapitre VII

Exercice 05 : UPCASE_IT

| | |
|---|---------------|
|  | Exercice : 05 |
| Voir en grand | |
| Dossier de rendu : <i>ex05/</i> | |
| Fichiers à rendre : upcase_it.rb | |
| Fonctions Autorisées : Toutes | |
| Remarques : n/a | |

- Créez un script `upcase_it.rb`.
- Ce script va d'abord demander un mot à l'utilisateur, puis l'affichera simplement en majuscules.
- Par exemple :


```
?> ./upcase_it.rb
Give me a word : banana
BANANA
?>
```



Google upcase.

Chapitre VIII

Exercice 06 : age

| | |
|---|---------------|
|  | Exercice : 06 |
| Recupérer et modifier un nombre | |
| Dossier de rendu : <i>ex06/</i> | |
| Fichiers à rendre : age.rb | |
| Fonctions Autorisées : Toutes | |
| Remarques : n/a | |

- Créez un script **age.rb** qui demande à l'utilisateur d'entrer son âge, puis affiche quel âge aura l'utilisateur dans 10 ans, 20 ans et 30 ans.

```
?> ./age.rb
Please tell me your age : 15
You are currently 15 years old.
In 10 years, you'll be 25 years old.
In 20 years, you'll be 35 years old.
In 30 years, you'll be 45 years old.
?>
```



Google `string to_i`.