



Initiation à la programmation en Ruby - 42

Jour 05

Staff 42 bocal@42.fr

Résumé: Ce document est le sujet du jour 05 de la piscine d'initiation à la programmation en Ruby.

Table des matières

I	Consignes	2
II	Exercice 00 : free_range	3
III	Exercice 01 : hello_wifilles	4
IV	Exercice 02 : upcase_it	5
V	Exercice 03 : downcase_all	6
VI	Exercice 04 : greetings_for_all	7
VII	Exercice 05 : methods_everywhere	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 : free_range

	Exercice : 00
D'ici à là, il n'y a qu'un pas!	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : free_range.rb	
Fonctions Autorisées : Toutes	
Remarques : n/a	

- Créez un script **free_range.rb** qui prend deux paramètres.
- Ces deux paramètres seront deux nombres.
- Vous devez construire un array contenant toutes les valeurs entre ces deux nombres en utilisant un range. Vous afficherez ensuite l'array avec la méthode `p`.
- Si le nombre de paramètres est différent de 2, vous afficherez 'none' suivi d'un retour à la ligne.


```
?> ./free_range.rb | cat -e
none$
?> ./free_range.rb 10 14 | cat -e
[10, 11, 12, 13, 14]$
?>
```



Google range.

Chapitre III

Exercice 01 : hello_wifilles

	Exercice : 01
On découvre les méthodes!	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : hello_wifilles.rb	
Fonctions Autorisées : Toutes	
Remarques : n/a	

- Créez un script `hello_wifilles.rb`
- Ce script va contenir une méthode `hello`. Cette méthode va afficher “Salut les wifilles!”.
- Après avoir défini votre méthode, vous allez la tester en l'appelant dans votre script. Comme dans l'exemple ci-dessous, sauf qu'on a caché la définition de la méthode.

```
?> cat hello_wifilles.rb
#votre definition de methode


hello()
?> ./hello_wifilles.rb
Salut les wifilles !
?>
```



Cherchez "définition d'une méthode en Ruby".

Chapitre IV

Exercice 02 : upcase_it

	Exercice : 02
Le retour d'upcase!	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : upcase_it.rb	
Fonctions Autorisées : Toutes	
Remarques : n/a	


- Créez un script **upcase_it.rb**. (encore lui!)
- Vous devez définir dans ce script une méthode. Cette méthode s'appelle **upcase_it**.
- La méthode **upcase_it** prend une chaîne de caractère comme argument. Elle doit retourner cette chaîne de caractère en majuscules.
- Testez la méthode en l'appelant dans votre script. Dans l'exemple ci-dessous, on teste avec "coucou" :

```
?> cat upcase_it.rb
# votre definition de methode

puts upcase_it(`coucou`)
?> ./upcase_it.rb
COUCOU
?>
```

Chapitre V

Exercice 03 : lowercase_all


	Exercice : 03
Baladons-nous dans l'array	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : lowercase_all.rb	
Fonctions Autorisées : Toutes	
Remarques : n/a	

- Créez un script **lowercase_all.rb**.
- Vous devez définir dans ce script une méthode. Cette méthode s'appelle **lowercase_it**.
- La méthode **lowercase_it** prend une chaîne de caractère comme argument. Elle doit retourner cette chaîne de caractère en minuscules.
- Vous appliquerez cette méthode, et afficherez son retour, sur les paramètres du script.
- S'il n'y a aucun paramètre, affichez none suivi d'un retour à la ligne.

```
?> ./lowercase_all.rb
none
?> ./lowercase_all.rb "HELLO WORLD" "J'ai bien compris les Tableaux !"
hello world
j'ai bien compris les tableaux !
?>
```

Chapitre VI

Exercice 04 : greetings_for_all

	Exercice : 04
On dit bonjour à la dame!	
Dossier de rendu : <i>ex04/</i>	
Fichiers à rendre : greetings_for_all.rb	
Fonctions Autorisées : Toutes	
Remarques : n/a	

- Créez un script `greetings_for_all.rb` qui ne prend pas de paramètre.
- Créez à l'intérieur une méthode `greetings` qui prend un nom en paramètre et affiche un message de bienvenue avec ce nom.
- Si la méthode est appelée sans argument, son paramètre par défaut sera “noble inconnue”.
- Si la méthode est appelée avec un argument qui n'est pas une chaîne de caractères, un message d'erreur devra être affiché à la place du message de bienvenue.
- Ainsi le script suivant :

```
?> cat greetings_for_all.rb | cat -e
# your method definition here

greetings('lucie')
greetings()
greetings(22)
```

aura la sortie :


```
?> ./greetings_for_all.rb | cat -e
Hello, lucie.$
Hello, noble inconnue.$
Erreur ! Ce n'etait pas un nom.$
?>
```




Google paramètre par défaut, méthode `is_a`.

Chapitre VII

Exercice 05 : `methods_everywhere`

	Exercice : 05
Des méthodes, partout !	
Dossier de rendu : <code>ex05/</code>	
Fichiers à rendre : <code>methods_everywhere.rb</code>	
Fonctions Autorisées : Toutes	
Remarques : n/a	

- Créez un script `methods_everywhere.rb` qui prend des paramètres.
- Vous devez créer deux méthodes différentes dans ce script :
- La méthode `retrecit` prend une chaîne de caractères en paramètre et affiche les huit premiers caractères de cette chaîne.



Utilisez les `slices`

- La méthode `agrandit` prend une chaîne de caractères en paramètre et la complète par des 'Z' pour qu'elle ait huit caractères en tout. Elle affiche ensuite la chaîne.



Comme pour les arrays, on peut ajouter des caractères à une chaîne avec l'opérateur «

- Pour chaque argument du script : si l'argument fait plus de 8 caractères, vous appelez la méthode `retrecit` dessus; si l'argument fait moins de 8 caractères, vous appelez la méthode `agrandit` dessus; et si l'argument fait 8 caractères, vous l'affichez directement suivi d'un retour à la ligne.
- Si le nombre de paramètres est inférieur à 1, vous afficherez 'none' suivi d'un retour

à la ligne.

```
?> ./methods_everywhere.rb | cat -e
none$
?> ./methods_everywhere.rb 'lol' 'agreablement' 'biquette' | cat -e
lolZZZZZ$
agreable$
biquette$
?>
```