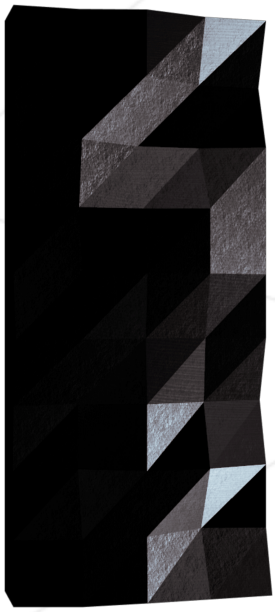

Initiation à la programmation en Ruby - 42

Jour 02

Staff 42 bocal@42.fr

Résumé: Ce document est le sujet du jour 02 de la piscine d'initiation à la programmation en Ruby.

Table des matières

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 formattage 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 : calculette

	Exercice : 00
Faire des petites opérations	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `calculette.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `calculette.rb`.
- Votre script va demander 2 nombres à l'utilisateur.
- Vous devez stocker ces nombres sous forme numérique dans deux variables.
- Vous devez ensuite afficher le résultat de leur addition, soustraction, division et multiplication.

```
?> ./calculette.rb
Donne moi un premier nombre : 10
Donne moi un deuxieme nombre : 2
Merci !
10 + 2 = 12
10 - 2 = 8
10 / 2 = 5
10 * 2 = 20
?>
```

Chapitre III

Exercice 01 : Des paramètres

	Exercice : 01
Oui, mais combien ?	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `parameters.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `parameters.rb`.
- Le script va afficher le nombre de paramètres qui lui sont passés, suivi d'un retour à la ligne.

```
?> ./parameters.rb
Nombre de parametres : 0.
?> ./parameters.rb "initiation"
Nombre de parametres : 1.
?> ./parameters "c'est" "la" "folie" "y'en a" "partout !"
Nombre de parametres : 5.
?>
```

`Google ARGV, array size.`

Chapitre IV

Exercice 02 : aff_first_param

	Exercice : 02
Afficher un paramètre	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `aff_first_param.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `aff_first_param.rb`.
- Le script affiche la première chaîne de caractères passée en paramètre suivie d'un retour à la ligne.
- S'il n'y a aucun paramètre, affichez `none` suivi d'un retour à la ligne.

```
?> ./aff_first_param.rb | cat -e
none$
?> ./aff_first_param.rb "Code Ninja" "Numerique" "42" | cat -e
Code Ninja$
?>
```

`Cherchez comment utiliser les conditions if.`

Chapitre V

Exercice 03 : UPCASE_IT

	Exercice : 03
Voir en grand	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `upcase_it.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `upcase_it.rb` qui prend une chaîne de caractères en paramètre.
- Le script doit afficher la chaîne de caractères en majuscules suivie d'un retour à la ligne.
- Si le nombre de paramètres est différent de 1, affichez `none` suivi d'un retour à la ligne.

```
?> ./upcase_it.rb | cat -e
none$
?> ./upcase_it.rb "initiation" | cat -e
INITIATION$
?> ./upcase_it.rb 'CeT eXeRcIcE eSt AsSeZ fAcIlE !' | cat -e
CET EXERCICE EST ASSEZ FACILE !$
?>
```

Chapitre VI

Exercice 04 : downcase_it

	Exercice : 04
Voir plus petit	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `downcase_it.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `downcase_it.rb` qui prend une chaîne de caractères en paramètre.
- Le script doit afficher la chaîne de caractères en minuscules suivie d'un retour à la ligne.
- Si le nombre de paramètres est différent de 1, affichez `none` suivi d'un retour à la ligne.

```
?> ./downcase_it.rb | cat -e
none$
?> ./downcase_it.rb "LUCIOLE" | cat -e
luciole$
?> ./downcase_it.rb 'CeT eXeRcIcE eSt AsSeZ fAcIlE !' | cat -e
cet exercice est assez facile !$
?>
```

`Cet exercice ne devrait pas vous prendre plus de 10 secondes.`

Chapitre VII

Exercice 05 : aff_rev_params

	Exercice : 05
Afficher un tableau à l'envers	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `aff_rev_params.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `aff_rev_params.rb`.
- Lorsqu'on l'exécute, le script affiche toutes les chaînes de caractères passées en paramètre, suivies d'un retour à la ligne et dans l'ordre inverse.
- S'il y a moins de deux paramètres, affichez `none` suivi d'un retour à la ligne.

```
?> ./aff_rev_params.rb | cat -e
none$
?> ./aff_rev_params.rb "coucou" | cat -e
none$
?> ./aff_rev_params.rb "Wi-filles" "piscine" "coucou la" | cat -e
coucou la$
piscine$
Wi-filles$
?>
```

`Google array reverse.`

Chapitre VIII

Exercice 06 : scan_it

	Exercice : 06
Scanner un texte	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `scan_it.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `scan_it.rb` qui prend deux paramètres.
- Le premier paramètre est un mot clé à chercher dans une chaîne.
- Le deuxième paramètre est la chaîne à parcourir.
- Lorsqu'on l'exécute, le programme affiche le nombre de fois où on trouve le mot-clé dans la chaîne.
- Si le nombre de paramètres est différent de 2 ou que la première chaîne n'apparaît pas dans la deuxième, affichez `none` suivi d'un retour à la ligne.

```
?> ./scan_it.rb | cat -e
none$
?> ./scan_it.rb "les" | cat -e
none$
?> ./scan_it.rb "les" "les exercices du J02 ne sont pas les plus difficiles" | cat -e
3$
?>
```

`Google "Ruby scan method".`