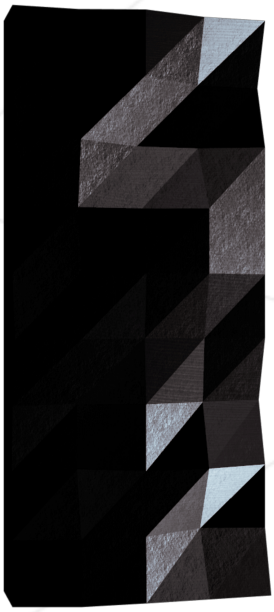

Initiation à la programmation en Ruby - 42

Jour 03

Staff 42 bocal@42.fr

Résumé: Ce document est le sujet du jour 03 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 : comparaison

	Exercice : 00
Identifier un paramètre	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `parameter_matching.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `parameter_matching.rb`.
- Ce script doit, si un paramètre est passé en argument, demander à l'utilisateur d'entrer un mot.
- Si le mot entré par l'utilisateur est le même que celui passé en paramètre, le script affichera "Good job !", sinon il affichera "Nope, sorry..." - suivis d'un retour à la ligne.
- Si le nombre de paramètres passés au script est différent de 1, il affichera "none" suivi d'un retour à la ligne.

```
?> ./parameter_matching.rb
none
?> ./parameter_matching.rb "Hello"
What was the parameter ? Bonjour
Nope, sorry...
?> ./parameter_matching.rb "Hello"
What was the parameter ? Hello
Good job !
?>
```

`Utilisez une (ou plusieurs ? :) ) structure if ... else ...`

Chapitre III

Exercice 01 : to25

	Exercice : 01
Allons jusqu'à 25 !	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `to25.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `to25.rb` qui prend un paramètre.
- Ce paramètre est un nombre, vous le stockerez dans une variable sous forme numérique.
- Vous allez ensuite faire une boucle qui affiche tous les nombres, du nombre fourni jusqu'à 25.
- Si le nombre de paramètres est différent de 1, vous afficherez "none" suivi d'un retour à la ligne.
- Si le paramètre est supérieur à 25, vous afficherez "Erreur" suivi d'un retour à la ligne.

```
?> ./to25.rb 8 9 10| cat -e
none$
?> ./to25.rb 20 | cat -e
Dans la boucle, ma variable vaut 20.$
Dans la boucle, ma variable vaut 21.$
Dans la boucle, ma variable vaut 22.$
Dans la boucle, ma variable vaut 23.$
Dans la boucle, ma variable vaut 24.$
Dans la boucle, ma variable vaut 25.$
?>
```

`While.`

Chapitre IV

Exercice 02 : table_mult

	Exercice : 02
Le retour des tables de multiplication	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `table_mult.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `table_mult.rb` qui prend un paramètre.
- Ce paramètre est la table de multiplication que vous allez afficher. (Par exemple, si le paramètre est 2, il faut afficher la table de 2)
- Si le nombre de paramètres est différent de 1, vous afficherez "none" suivi d'un retour à la ligne.

```
?> ./table_mult.rb | cat -e
none$
?> ./table_mult.rb 8 | cat -e
0 x 8 = 0$
1 x 8 = 8$
2 x 8 = 16$
3 x 8 = 24$
4 x 8 = 32$
5 x 8 = 40$
6 x 8 = 48$
7 x 8 = 56$
8 x 8 = 64$
9 x 8 = 72$
?>
```

Chapitre V

Exercice 03 : i_got_that

	Exercice : 03
Do you got it ?	
Dossier de rendu : *ex*03/	
Fichiers à rendre : `i_got_that.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `i_got_that.rb`. Ce script doit contenir une boucle while qui accepte un input de l'utilisateur, écrit une phrase en retour, et s'arrête uniquement lorsque l'utilisateur a entré "STOP". Chaque tour de boucle doit accepter un input de l'utilisateur.

```
?> ./i_got_that.rb
What you gotta say ? : Hello
I got that ! Anything else ? : I like ponies
I got that ! Anything else ? : stop...
I got that ! Anything else ? : STOP
?>
```

`While, break.`

Chapitre VI

Exercice 04 : advanced mult

	Exercice : 04
Le retour du retour des tables de multiplication	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `advanced_mult.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `advanced_mult.rb` qui ne prend pas de paramètre.
- Si le nombre de paramètres est différent de 0, vous afficherez "none" suivi d'un retour à la ligne.
- Ce script va afficher toutes les tables de multiplication sous la forme suivante :

```
?> ./advanced_mult.rb "yolo" | cat -e
none$
?> ./advanced_mult.rb
Table de 0: 0 0 0 0 0 0 0 0 0 0 0
Table de 1: 0 1 2 3 4 5 6 7 8 9 10
Table de 2: 0 2 4 6 8 10 12 14 16 18 20
Table de 3: 0 3 6 9 12 15 18 21 24 27 30
Table de 4: 0 4 8 12 16 20 24 28 32 36 40
Table de 5: 0 5 10 15 20 25 30 35 40 45 50
Table de 6: 0 6 12 18 24 30 36 42 48 54 60
Table de 7: 0 7 14 21 28 35 42 49 56 63 70
Table de 8: 0 8 16 24 32 40 48 56 64 72 80
Table de 9: 0 9 18 27 36 45 54 63 72 81 90
Table de 10: 0 10 20 30 40 50 60 70 80 90 100
?>
```

- Vous n'avez le droit qu'à deux boucles while.