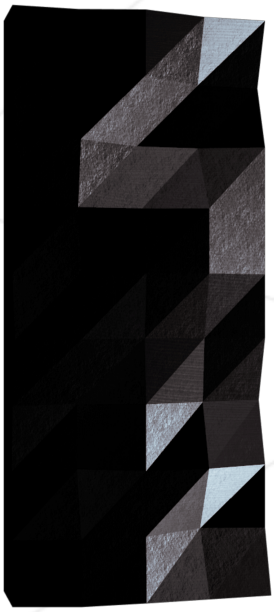

Initiation à la programmation - 42

Ruby - Jour 2

Staff 42 pedago@42.fr

Résumé: Ce document est le sujet du jour 2 du programme d'initiation à Ruby.

Table des matières

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices.
- Vous serez corrigés par les autres participants du programme d'initiation. C'est le "peer-correcting" de la pédagogie 42 !
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.
- Lisez attentivement les exemples. Les exercices pourraient bien requérir des choses qui y sont précisées, et non dans le sujet...

Chapitre II

Procédure de rendu : Git

- Votre rendu est collecté via un serveur distant. Cela signifie que vous devrez envoyer sur ce serveur votre repertoire de rendu avec vos exercices suivant une arborescence précise.
- Le logiciel utilisé pour ce faire s'appelle `git`.
- Git permet de faire beaucoup de choses, vous trouverez ci-dessous les principales commandes :
- `Git clone` vous permet de `cloner` un répertoire présent sur un serveur directement dans votre répertoire courant. Cela signifie que tous les fichiers présents que vous y aviez mis depuis un autre poste se retrouveront sur votre ordinateur et pourront être modifiés et réenvoyés avec les commandes git suivantes. Cette commande est aussi importante quand vous souhaitez vérifier ce que vous avez "pushé" sur le serveur, n'hésitez pas à en abuser.

```
?> git clone vogsphere@vogsphere.42.fr:piscine/truc/machin/creme
Cloning into 'piscine'[...]
?>
```

- `Git add` vous permet d'ajouter à une liste de fichiers surveillés un ou plusieurs fichiers/repertoires. Ils seront ajoutés en l'état ce qui veut dire que si vous les modifiez par la suite, il faudra les ajouter à nouveau pour mettre à jour la liste.

```
?> git add file directory
?>
```

- `Git commit -m` vous permet de "fixer" votre liste de fichiers surveillés afin de préparer leur envoi sur le serveur. Le message est obligatoire et permet d'avoir un historique des créations et modifications que vous avez effectué au fur et à mesure de votre travail.

```
?> git commit -m "Ajout des exercices X Y Z"
[master (root-commit) 4e8b2aa] Ajout des exercices X Y Z
3 file changed, 500 insertions(+)
create mode 100755 X Y Z
?>
```

- `Git push origin master`, la commande ultime, celle qui vous permet d'envoyer sur le serveur vos exercices qui y seront collectés par la Moulinette. Vérifiez bien son retour, le moindre message d'erreur signifie que vous n'avez rien pu envoyer et que par conséquent votre répertoire sur le serveur est vide (ce qui serait dommage pour votre note). Si vous avez le moindre doute, contactez un membre du staff 42.

```
?> git push origin master
[...]
X files written.
?>
```

L'adresse git de votre dépôt se trouve sur l'intranet et dans le fichier config du répertoire .git de votre dossier de rendu, c'est avec elle que vous pourrez "git clone".

Chapitre III

Préambule

Le jeu du Sirop selon `Perceval`, tiré de la série *Kaamelott* :

"Bon, j'vais vous apprendre les règles simplifiées, parce que les vraies règles, elles sont velues. Bon, le seul truc, c'est que normalement, ça se joue à trois. Mais c'est pas grave on va se débrouiller.

Le principe, c'est de faire des valeurs. Donc là, mettons, on est trois, il y a trois valeurs à distribuer. On va dire, sirop de huit, sirop de quatorze et sirop de vingt-et-un. Vous occupez pas des sirops tout de suite. Ce qu'il faut comprendre d'abord, c'est les valeurs. Si vous lancez une valeur en début de tour, mettons un sirop de huit, pour commencer petit, les autres ont le choix entre laisser filer la mise ou relancer un sirop de quatorze. On tourne dans le sens des valeurs. C'est pour ça, il faut bien comprendre le système des valeurs; après, ça va tout seul.

Bon alors mettons que j'ouvre avec un sirop de huit.
Si c'est vous qu'avez siroté au tour d'avant, ça tourne dans votre sens. Alors soit vous laissez filer, vous dites "file-sirop", soit vous vous sentez de relancer et vous annoncez un sirop de quatorze. Comme on a commencé les annonces, le second joueur a pas le droit de laisser filer. Vous pouvez soit relancer un sirop de vingt-et-un, soit vous abandonnez le tour et vous dites "couche-sirop" ou "sirop Jeannot", ça dépend des régions. Et après, soit on fait la partie soit je fais un "contre-sirop" ! Et à partir de là, sirop de pomme sur vingt-et-un donc on fait la partie en quatre tours jusqu'à qu'il y en ait un qui sirote.

À la gagne, il n'y a que trois possibilités : soit vous faites votre sirop de huit, vous dites "beau sirop" et on recompte, soit vous faites votre sirop de quatorze, vous dites "beau sirop, sirop gagnant" et on vous rajoute la moitié, soit vous faites votre sirop de vingt-et-un et vous dites "beau sirop, mi-sirop, siroté, gagne-sirop, sirop-grelot, passe-montagne, sirop au bon goût".

Normalement ça se joue avec des cartes mais si vous avez que des dés, vous pouvez aussi jouer avec des dés puisque ce qui compte c'est les valeurs."

Au moins un des exercices suivants n'a aucun rapport avec le jeu du Sirop.

Chapitre IV

Exercice 00 : to_n

	Exercice : 00
Compter jusqu'à n	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `to_n.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `to_n.rb` qui prend deux nombres en paramètre et affiche tous les nombres, du premier argument jusqu'au deuxième. Si le nombre de paramètres est différent de 2, ou si le premier argument est plus grand que le second, vous afficherez "none" suivi d'un retour à la ligne.

```
?> ./to_n.rb 8 9 10| cat -e
none$
?> ./to_n.rb 24 23 | cat -e
none$
?> ./to_n.rb 20 25 | cat -e
20$
21$
22$
23$
24$
25$
?>
```

Chapitre V

Exercice 01 : table_mult

	Exercice : 01
Le retour des tables de multiplication	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `table_mult.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `table_mult.rb` qui prend un nombre en paramètre, et affiche la table de multiplication de ce nombre. 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 VI

Exercice 02 : puts_each

	Exercice : 02
Baladons-nous dans l'array	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `puts_each.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `puts_each.rb` qui déclare un array de valeur [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], et utilise la methode each pour itérer sur l'array et afficher chaque valeur.

```
?> ./puts_each.rb | cat -e
1$
2$
3$
4$
5$
6$
7$
8$
9$
10$
?>
```

Chapitre VII

Exercice 03 : strings_are_arrays

	Exercice : 03
Mieux comprendre les chaînes de caractères	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `strings_are_arrays.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `strings_are_arrays.rb` qui prend en paramètre une chaîne de caractères. Lorsqu'on l'exécute, le script affiche "z" pour chaque caractère "z" se trouvant dans la chaîne passée en paramètre, le tout suivi d'un retour à la ligne.

- Si le nombre de paramètres est différent de 1, ou s'il n'y a aucun caractère "z" dans la chaîne, affichez `none` suivi d'un retour à la ligne.

```
?> ./strings_are_arrays.rb | cat -e
none$
?> ./strings_are_arrays.rb "Le caractere recherche ne se trouve pas dans cette chaine de
 caracteres" | cat -e
none$
?> ./strings_are_arrays.rb "z" | cat -e
z$
?> ./strings_are_arrays.rb "Zaz visite le zoo avec Zazie" | cat -e
zzz$
?>
```

Chapitre VIII

Exercice 04 : play_with_arrays

	Exercice : 04
Manipuler des arrays	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `play_with_arrays.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `play_with_arrays.rb` qui itére sur les nombres passés en paramètre et construit un nouvel array qui est le résultat de l'addition de la valeur 2 à chaque valeur passée en paramètre. Affichez l'array résultant en utilisant la méthode p plutôt que puts.

```
?> ./play_with_arrays.rb 2 8 9 48 8 22 -12 2 | cat -e
[4, 10, 11, 50, 10, 24, -10, 4]$
?>
```

Chapitre IX

Exercice 05 : play_with_arrays++

	Exercice : 05
Manipuler des arrays, encore	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `play_with_arrays.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Reprenez le script précédent, mais cette fois vous ne traiterez que les valeurs supérieures à 5 passées en paramètre.

```
?> ./play_with_arrays.rb 2 8 9 48 8 22 -12 2 | cat -e
[10, 11, 50, 10, 24]$
?>
```

Chapitre X

Exercice 06 : play_with_arrays+=2

	Exercice : 06
Manipuler des arrays, toujours !	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `play_with_arrays.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Reprenez le script précédent, mais cette fois vous n'afficherez plus les doublons à la sortie, et les valeurs seront triées par ordre croissant. Attention, vous ne devez pas explicitement retirer des valeurs de vos arrays.

```
?> ./play_with_arrays.rb 2 8 9 48 8 22 -12 2 | cat -e
[10, 11, 24, 50]$
?>
```

Chapitre XI

Exercice 07 : count_it

	Exercice : 07
Compter et mesurer des paramètres	
Dossier de rendu : $ex07/$	
Fichiers à rendre : `count_it.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `count_it.rb` qui, lorsqu'on l'exécute, affiche "`parametres:`" puis le nombre de paramètres passés en argument suivi d'un retour à la ligne, puis chaque paramètre et sa taille suivi d'un retour à la ligne. S'il n'y a aucun paramètre, affichez `none` suivi d'un retour à la ligne.

```
?> ./count_it.rb | cat -e
none$
?> ./count_it.rb "Game" "of" "Thrones" | cat -e
parametres: 3$
Game: 4$
of: 2$
Thrones: 7$
?>
```

Chapitre XII

Exercice 08 : append_it

	Exercice : 08
Modifier des chaînes de caractères	
Dossier de rendu : $ex08/$	
Fichiers à rendre : `append_it.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `append_it.rb` qui affiche les paramétres passés en argument, un à un, en retirant la dernière lettre de chaque paramétre et en affichant “isme” à la place. Si le paramétre termine deja par “isme”, on passe au suivant, il n’est pas affiché. S’il n’y a aucun paramètre, affichez `none` suivi d’un retour à la ligne.

```
?> ./append_it.rb | cat -e
none$
?> ./append_it.rb "parallele" "egoisme" "morale" | cat -e
parallelisme$
moralisme$
?>
```