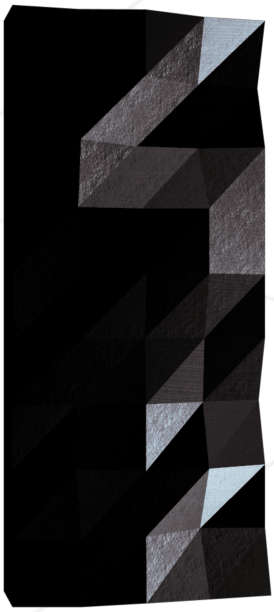

Initiation à la programmation - 42

Jour 02

Staff 42 bocal@42.fr

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

Table des matières

Chapitre I

Consignes

- La procédure de correction se déroulera durant la dernière heure de la journée. Chaque personne corrigera une autre personne selon le model de peer-correcting.
- Vous avez une question ? Demandez à votre voisine de droite. Sinon, essayez avec votre voisine de gauche.
- Lisez attentivement les exemples. Les exercices pourraient bien requérir des choses qui y sont précisées, et non dans le sujet...
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.

Chapitre II

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 III

Exercice 00 : puts_each

	Exercice : 00
Baladons-nous dans l'array	
Dossier de rendu : $ex00/$	
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$
?>
```

`Google array, each.`

Chapitre IV

Exercice 01 : aff_rev_params

	Exercice : 01
Afficher un tableau à l'envers	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `aff_rev_params.rb`	
Fonctions Autorisées : `Toutes`	
Remarques : `n/a`	

- Créez un script `aff_rev_params.rb` qui, lorsqu'on l'exécute, 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 ARGV, array, reverse.`

Chapitre V

Exercice 02 : i_got_that

	Exercice : 02
Do you got it ?	
Dossier de rendu : *ex*02/	
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
?>
```

`Google while, break.`

Chapitre VI

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$
?>
```

Les chaînes de caractères sont aussi composées de cases. Essayez !

Chapitre VII

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 un array de nombres (que vous définirez) et construit un nouvel array qui est le résultat de l'addition de la valeur 2 à chaque valeur de l'array d'origine. Vous devez avoir deux arrays à la fin du programme, celui d'origine et le nouveau que vous avez créé. Affichez les deux arrays à l'ecran en utilisant la méthode p plutôt que puts.
- Par exemple, si votre array d'origine est [2, 8, 9, 48, 8, 22, -12, 2], vous aurez la sortie suivante :

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

`Google p method in ruby, array each`

Chapitre VIII

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 de l'array d'origine.
- Par exemple, si votre array d'origine est [2, 8, 9, 48, 8, 22, -12, 2], vous aurez la sortie suivante :

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

`Google p method in ruby, array each`

Chapitre IX

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. Attention, vous ne devez pas explicitement retirer des valeurs de vos arrays.
- Par exemple, si votre array d'origine est [2, 8, 9, 48, 8, 22, -12, 2], vous aurez la sortie suivante :

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

`Google array, uniq`

Chapitre X

Exercice 07 : count_it

	Exercice : 07
Compter et mesurer des paramètres	
Dossier de rendu : *ex*07/	
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$
?>
```

`Google size.`

`Respectez strictement le format de sortie indiqué dans l'exemple.`

Chapitre XI

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$
?>
```

`Google match.`

`Respectez strictement le format de sortie indiqué dans l’exemple.`