

# Initiation à la programmation - 42 Jour 04

Staff 42 bocal@42.fr

Résumé: Ce document est le sujet du jour 04 de la piscine d'initiation à la programmation.

# Table des matières

1	Consignes	2
II	Exercice 00 : count_it	3
III	Exercice 01 : strings_are_arrays	4
IV	Exercice 02 : play_with_arrays	5
V	Exercice 03 : play_with_arrays++	6
VI	Exercice 04 : play_with_arrays+=2	7
VII	Exercice 05: append_it	8

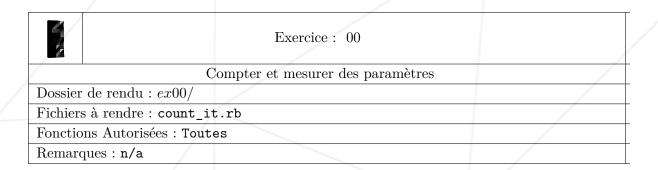
#### Chapitre I

#### 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: count\_it



- Créez un script count\_it.rb.
- Le script va afficher "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.

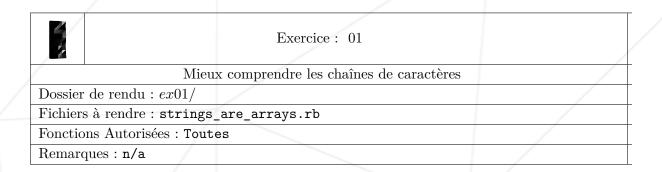
```
?> ./count_it.rb | cat -e
none$
?> ./count_it.rb "Game" "of" "Thrones" | cat -e
parametres: 3$
Game: 4$
of: 2$
Thrones: 7$
?>
```



Cette fois vous n'allez pas utiliser les boucles while, mais la  $m\acute{e}$ thode each.

#### Chapitre III

## Exercice 01: strings\_are\_arrays



- 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.

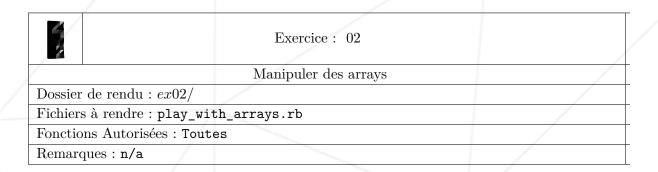
```
?> ./strings_are_arrays.rb | cat -e
none$
?> ./strings_are_arrays.rb "Le caractere recherche ne se trouve pas dans cette chaine de
caracteres" | cat -e
none$
?> ./strings_are_arrays.rb "z" | cat -e
z$
?> ./strings_are_arrays.rb "Zaz visite le zoo avec Zazie" | cat -e
zzz$
?>
```



Les chaînes de caractères sont aussi composées de cases, comme les arrays. Essayez!

#### Chapitre IV

#### Exercice 02: play\_with\_arrays



- Créez un script play\_with\_arrays.rb.
- Vous allez d'abord définir un array de nombres.
- Puis votre script va itérer sur cet array et construire un nouvel array en ajoutant 2 à chaque valeur de l'array d'origine.
- Vous devez donc avoir deux arrays dans votre programme, celui d'origine et le nouveau que vous avez créé.
- A la fin, 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 "méthode p en ruby", each

## Chapitre V

# Exercice 03: play\_with\_arrays++

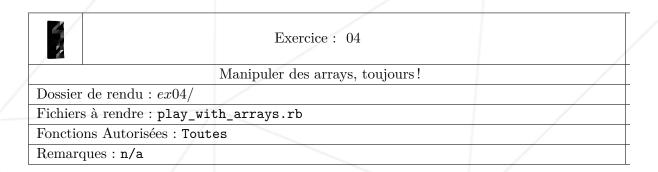
	Exercice: 03	
	Manipuler des arrays, encore	
Dossier de rendu : $ex03/$		
Fichiers à rendre : play_with_arrays.rb		
Fonctions Autorisées : Toutes		/
Remarques : n/a		

- 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]$
?>
```

#### Chapitre VI

#### Exercice 04: play\_with\_arrays+=2



- 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 :

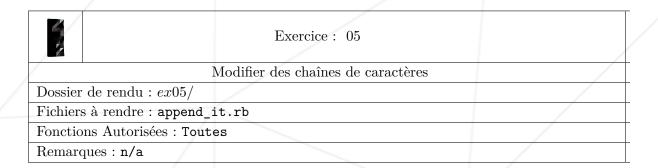
```
?> ./play_with_arrays.rb | cat -e
[2, 8, 9, 48, 22, -12]$
[10, 11, 50, 24]$
?>
```



Uniq.

# Chapitre VII

# Exercice 05: append\_it



- Créez un script append\_it.rb.
- Le script va afficher 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 paramètre 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$
?>
```



Match.