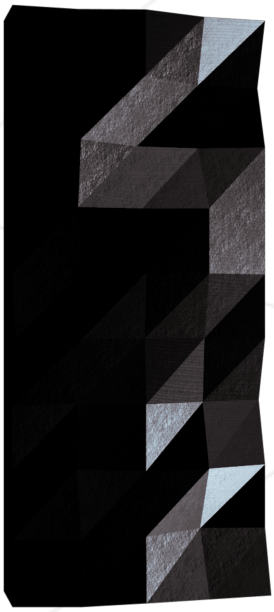

Ruby - Cellule 06

Méthodes & scope

Staff Pédago 42 pedago@42.fr

Résumé: Dans cette cellule, nous voyons comment utiliser les méthodes et scope.

Table des matières

I	Préambule	2
II	Consignes générales	4
III	Exercice 00 : hello_all	5
IV	Exercice 01 : upcase_it	6
V	Exercice 02 : downcase_all	7
VI	Exercice 03 : greetings_for_all	8
VII	Exercice 04 : methods_everywhere	10
VIII	Exercice 05 : scope_that	12

Chapitre I

Préambule

`Recette du kouign-amann`

Préparation : 4h 00 min
Cuisson : 35 minutes

Ingredients

250 g de farine
200 g de beurre
200 g de sucre en poudre
10 g de levure fraîche
10 cl d' eau
2 pincées de sel

Instructions

Pensez tout d'abord à sortir le beurre demi-sel ou salé* du réfrigérateur afin qu'il ramollisse.
Commencez par réaliser la pâte du kouign-amann : mélangez la levure de boulanger fraîche (surtout pas de la levure chimique !) et 3 cuillères à soupe d'eau tiède non brûlante dans une tasse, puis, dans un saladier, mélangez la farine de blé et ajoutez 2 pincées de sel (attention, le sel et la levure boulangère ne doivent pas être en contact, ceci risquerait de tuer la levure et empêcher votre pâte de gonfler).
Formez un puits et versez-y votre mélange de levure et 10 cl d'eau.
Farinez votre plan de travail et travaillez votre pâte à kouign-amann jusqu'à l'obtention d'une pâte souple.
Laissez la pâte reposer en boule dans le saladier à température ambiante pendant 3 h (étape 1 du schéma).
Au bout des 3 heures de repos, la pâte aura triplé de volume : sur votre plan de travail fariné, à l'aide d'un rouleau à pâtisserie, abaissez-la pâte (cette étape peut-être assez longue car la pâte est relativement élastique) et donnez lui une forme rectangulaire ou carrée d'environ 1 cm de hauteur (étape 2 du schéma) : étalez le beurre demi-sel bien mou au pinceau et saupoudrez de sucre en poudre. Prenez soin de ne pas étaler le beurre et le sucre sur les rebords de la pâte et de laisser un espace de 3 cm environ tout autour (étape 3 du schéma).

Repliez la pâte à kouign-amann en 3 dans la longueur (étape 4 du schéma) puis de nouveau en 3 dans la largeur (étape 5 du schéma), à la manière d'une pâte feuilletée, le but étant de bien "emprisonner" le beurre et le sucre.
Avec votre rouleau à pâtisserie, étalez de nouveau la pâte très délicatement en faisant attention à ce que le beurre ne ressorte pas (étape 6 du schéma).
Repliez la pâte encore une fois en 3 dans la longueur puis dans la largeur (étape 7 et 8 du schéma).
Tassez votre pâton dans un moule beurré et laissez le reposer pendant une demie-heure.
Avec un couteau, tracez un quadrillage sur la pâte et parsemez quelques morceaux de beurre supplémentaires.
Préchauffez le four thermostat 7 (210°).
Puis, faites cuire le kouign-amann pendant 35 mn environ.
Une fois votre kouign-amann cuit, sortez-le du four et patientez un quart d'heure avant de procéder au démoulage.
Sapoudrez votre kouign-amann avec un peu de sucre en poudre et dégustez-le encore tiède !

Astuce du chef :
Suite à un commentaire d'un internaute, je vous déconseille d'utiliser du beurre aux cristaux de sel de Guérande ou à base de gros sel : les gros grains de sel ont tendance à déchirer votre pâte lors de l'étalage du beurre faisant ainsi fuir le beurre pendant la cuisson !

Chapitre II

Consignes générales

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les jours de cette Piscine.

- Seul ce sujet sert de référence : ne vous fiez pas aux bruits de couloir.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vos exercices seront évalués par vos camarades de Piscine.
- Vous ne devez laisser aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices dans votre répertoire lors de la peer-évaluation.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Toutes les réponses à vos questions techniques se trouvent dans les `man` ou sur Internet.
- Pensez à discuter sur le forum Piscine de votre Intra et sur Slack !
- Lisez attentivement les exemples car ils peuvent vous permettre d'identifier un travail à réaliser qui n'est pas précisé dans le sujet à première vue.
- Réfléchissez. Par pitié, par Thor, par Odin !

Chapitre III

Exercice 00 : hello_all

	Exercice : 00
On découvre les méthodes !	
Dossier de rendu : *ex*00/	
Fichiers à rendre : `hello_all.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `hello_all.rb`.
- Ce programme doit être exécutable.
- Ce programme va contenir une méthode hello. Cette méthode va afficher “Salut tout le monde !”.
- Après avoir défini votre méthode, vous allez la tester en l’appelant dans votre programme. Comme dans l’exemple ci-dessous, sauf qu’on a caché la définition de la méthode.

```
?> cat hello_all.rb
#votre definition de methode

hello()
?> ./hello_all.rb
Salut tout le monde !
?>
```

Cherchez "définition d’une méthode en Ruby".

Chapitre IV

Exercice 01 : upcase_it

	Exercice : 01
Le retour d'upcase !	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `upcase_it.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `upcase_it.rb` (encore lui !)
- Ce programme doit être exécutable.
- Vous devez définir dans ce programme une méthode. Cette méthode s'appelle `upcase_it`.
- La méthode `upcase_it` prend une chaîne de caractère comme argument. Elle doit retourner cette chaîne de caractère en majuscules.
- Testez la méthode en l'appelant dans votre programme. Dans l'exemple ci-dessous, on teste avec "coucou" :

```
?> cat upcase_it.rb
# votre definition de methode

puts upcase_it("coucou")
?> ./upcase_it.rb
COUCOU
?>
```

Chapitre V

Exercice 02 : downcase_all

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

- Créez un programme `downcase_all.rb`.
- Ce programme doit être exécutable.
- Vous devez définir dans ce programme une méthode. Cette méthode s'appelle `downcase_it`.
- La méthode `downcase_it` prend une chaîne de caractère comme argument. Elle doit retourner cette chaîne de caractère en minuscules.
- Vous appliquerez cette méthode, et afficherez son retour, sur les paramètres du programme.
- S'il n'y a aucun paramètre, affichez none suivi d'un retour à la ligne.

```
?> ./downcase_all.rb
none
?> ./downcase_all.rb "HELLO WORLD" "J'ai bien compris les Tableaux !"
hello world
j'ai bien compris les tableaux !
?>
```

Chapitre VI

Exercice 03 : greetings_for_all

	Exercice : 03
On dit bonjour à la dame !	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `greetings_for_all.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `greetings_for_all.rb` qui ne prend pas de paramètre.
- Ce programme doit être exécutable.
- Créez à l'intérieur une méthode `greetings` qui prend un nom en paramètre et affiche un message de bienvenue avec ce nom.
- Si la méthode est appelée sans argument, son paramètre par défaut sera "noble inconnue".
- Si la méthode est appelée avec un argument qui n'est pas une chaîne de caractères, un message d'erreur devra être affiché à la place du message de bienvenue.
- Ainsi le programme suivant :

```
?> cat greetings_for_all.rb | cat -e
# your method definition here

greetings('lucie')
greetings()
greetings(22)
```

aura la sortie :

```
?> ./greetings_for_all.rb | cat -e
Hello, lucie.$
Hello, noble inconnue.$
Erreur ! Ce n'etait pas un nom.$
?>
```

`Google paramètre par défaut, méthode is_a.`

Chapitre VII

Exercice 04 : methods_everywhere

	Exercice : 04
Des méthodes, partout !	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `methods_everywhere.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `methods_everywhere.rb` qui prend des paramètres.
- Ce programme doit être exécutable.
- Vous devez créer deux méthodes différentes dans ce programme :
- La méthode `retrecit` prend une chaîne de caractères en paramètre et affiche les huit premiers caractères de cette chaîne.

`Utilisez les slices.`

- La méthode `agrandit` prend une chaîne de caractères en paramètre et la complète par des 'Z' pour qu'elle ait huit caract 'eres en tout. Elle affiche ensuite la chaîne.

`Comme pour les arrays, on peut ajouter des caractères à une chaîne avec l'opérateur «`

- Pour chaque argument du programme : si l'argument fait plus de 8 caractères, vous appelez la méthode `retrecit` dessus ; si l'argument fait moins de 8 caractères, vous appelez la méthode `agrandit` dessus ; et si l'argument fait 8 caractères, vous l'affichez directement suivi d'un retour à la ligne.
- Si le nombre de paramètres est inférieur à 1, vous afficherez `'none'` suivi d'un

retour à la ligne.

```
?> ./methods_everywhere.rb | cat -e
none$
?> ./methods_everywhere.rb 'lol' 'agreablement' 'biquette' | cat -e
lolZZZZZ$
agreable$
biquette$
?>
```

Chapitre VIII

Exercice 05 : scope_that

	Exercice : 05
Can't touch this !	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `scope_that.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `scope_that.rb` qui ne prend pas de paramètre.
- Ce programme doit être exécutable.
- Créez à l'intérieur une méthode `add_one` qui prend un paramètre et qui ajoute 1 au paramètre passé à la méthode.
- Initialisez une variable dans le corps du programme, affichez la, puis appelez la methode qui ajoute 1.
- Affichez a nouveau votre variable dans le corps du programme.
- Que constatez-vous ?