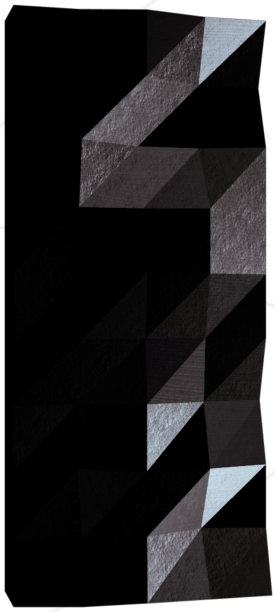

Ruby - Cellule 05

Tableaux & fonctions associées

Staff Pédago 42 pedago@42.fr

Résumé: Dans cette cellule, nous voyons comment utiliser les tableaux et fonctions associées.

Table des matières

Chapitre I

Préambule

`Recette de la pissaladière`

Préparation
TEMPS TOTAL : 1H15
Préparation : 30 min
Cuisson : 45 min

Etape 1
Dans une grande poêle profonde faire chauffer 5 cuillères à soupe d'huile d'olive.

Etape 2
Couper les oignons en rondelles.

Etape 3
Une fois que l'huile est chaude vous pouvez mettre les oignons dans la poêle, ajoutez du poivre, les herbes de Provence et le sucre en poudre. Il ne faut surtout pas mettre de sel car il y aura les anchois. Le sucre est indispensable pour enlever l'acidité de l'oignon.

Etape 4
Faites revenir les oignons jusqu'à que leur couleur soit légèrement jaune. Il ne faut pas qu'ils reviennent de trop car ils vont encore cuire dans le four.

Etape 5
Le secret de la Pissaladière est là. Il faut garder quelques filets d'anchois et mettre le reste dans la poêle avec les oignons. Les anchois vont fondre à la chaleur et se mélanger aux oignons. Si vous pouvez ajouter une cuillère à soupe de l'huile des anchois dans la préparation ce sera meilleur.

Etape 6
Vous pouvez étaler la pâte à pain sur une plaque de four huilée à l'huile d'olive.

Etape 7
Versez votre préparation sur la pâte et mettez des anchois et des olives pour la décoration.

Etape 8

Préchauffez votre four à 220°C et mettre ensuite la pissaladière.

Etape 9
Pour le temps de cuisson dès que la pâte à pain est cuite (voir aux bords de la pâte) vous pouvez sortir votre pissaladière.

Etape 10
Il faut laisser la pissaladière refroidir car elle se déguste froide.

Etape 11
Vous pouvez l'accompagner d'une salade frisée et d'un Bandol rosé c'est excellent.

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 : create_array

	Exercice : 00
Créer un array	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `create_array.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `create_array.rb`.
- Ce programme doit être exécutable.
- Vous allez définir un array de nombres.
- Vous allez afficher votre array à l'ecran :

```
?> ./create_array.rb | cat -e
[2, 8, 9, 48, 8, 22, -12, 2]$
?>
```

`print, puts, p.`

Chapitre IV

Exercice 01 : play_with_arrays

	Exercice : 01
Manipuler des arrays	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `play_with_arrays.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `play_with_arrays.rb`.
- Ce programme doit être exécutable.
- Vous allez d'abord définir un array de nombres.
- Puis votre programme 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 02 : play_with_arrays++

	Exercice : 02
Manipuler des arrays, encore	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `play_with_arrays.rb`	
Fonctions Autorisées : `Toutes`	

- Reprenez le programme 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 03 : play_with_arrays+=2

	Exercice : 03
Manipuler des arrays, toujours !	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `play_with_arrays.rb`	
Fonctions Autorisées : `Toutes`	

- Reprenez le programme 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 04 : Des paramètres

	Exercice : 04
Oui, mais combien ?	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `parameters.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `parameters.rb`.
- Ce programme doit être exécutable.
- Le programme va afficher le nombre de paramètres qui lui sont passés, suivi d'un retour à la ligne.

```
?> ./parameters.rb
Nombre de parametres : 0.
?> ./parameters.rb "initiation"
Nombre de parametres : 1.
?> ./parameters "c'est" "la" "folie" "y'en a" "partout !"
Nombre de parametres : 5.
?>
```

`Google ARGV, array size.`

Chapitre VIII

Exercice 05 : aff_first_param

	Exercice : 05
Afficher un paramètre	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `aff_first_param.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `aff_first_param.rb`.
- Ce programme doit être exécutable.
- Le programme affiche la première chaîne de caractères passée en paramètre suivie d'un retour à la ligne.
- S'il n'y a aucun paramètre, affichez none suivi d'un retour à la ligne.

```
?> ./aff_first_param.rb | cat -e
none$
?> ./aff_first_param.rb "Code Ninja" "Numerique" "42" | cat -e
Code Ninja$
?>
```

Cherchez comment utiliser les conditions if.

Chapitre IX

Exercice 06 : UPCASE_IT

	Exercice : 06
Voir en grand	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `upcase_it.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `upcase_it.rb` qui prend une chaîne de caractères en paramètre.
- Ce programme doit être exécutable.
- Le programme doit afficher la chaîne de caractères en majuscules suivie d'un retour à la ligne.
- Si le nombre de paramètres est différent de 1, affichez none suivi d'un retour à la ligne.

```
?> ./upcase_it.rb | cat -e
none$
?> ./upcase_it.rb "initiation" | cat -e
INITIATION$
?> ./upcase_it.rb 'CeT eXeRcIcE eSt AsSeZ fAcIlE !' | cat -e
CET EXERCICE EST ASSEZ FACILE !$
?>
```

Chapitre X

Exercice 07 : downcase_it

	Exercice : 07
Voir plus petit	
Dossier de rendu : *ex*07/	
Fichiers à rendre : `downcase_it.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `downcase_it.rb` qui prend une chaîne de caractères en paramètre.
- Ce programme doit être exécutable.
- Le programme doit afficher la chaîne de caractères en minuscules suivie d'un retour à la ligne.
- Si le nombre de paramètres est différent de 1, affichez none suivi d'un retour à la ligne.

```
?> ./downcase_it.rb | cat -e
none$
?> ./downcase_it.rb "LUCIOLE" | cat -e
luciole$
?> ./downcase_it.rb 'CeT eXeRcIcE eSt AsSeZ fAcIlE !' | cat -e
cet exercice est assez facile !$
?>
```

Cet exercice ne devrait pas vous prendre plus de 10 secondes.

Chapitre XI

Exercice 08 : aff_rev_params

	Exercice : 08
Afficher un tableau à l'envers	
Dossier de rendu : $ex08/$	
Fichiers à rendre : `aff_rev_params.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `aff_rev_params.rb`.
- Ce programme doit être exécutable.
- Lorsqu'on l'exécute, le programme 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 "Ruby" "piscine" "coucou la" | cat -e
coucou la$
piscine$
Ruby$
?>
```

`Google array reverse.`

Chapitre XII

Exercice 09 : scan_it

	Exercice : 09
Scanner un texte	
Dossier de rendu : *ex*09/	
Fichiers à rendre : `scan_it.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `scan_it.rb` qui prend deux paramètres.
- Le premier paramètre est un mot clé à chercher dans une chaîne.
- Le deuxième paramètre est la chaîne à parcourir.
- Ce programme doit être exécutable.
- Lorsqu'on l'exécute, le programme affiche le nombre de fois où on trouve le mot-clé dans la chaîne.
- Si le nombre de paramètres est différent de 2 ou que la première chaîne n'apparaît pas dans la deuxiéme, affichez none suivi d'un retour à la ligne.

```
?> ./scan_it.rb | cat -e
none$
?> ./scan_it.rb "les" | cat -e
none$
?> ./scan_it.rb "les" "les exercices du C05 ne sont pas les plus difficiles" | cat -e
3$ ?>
```

`Google "Ruby scan method".`

Chapitre XIII

Exercice 10 : comparaison

	Exercice : 10
Identifier un paramètre	
Dossier de rendu : $ex10/$	
Fichiers à rendre : `parameter_matching.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `parameter_matching.rb`.
- Ce programme doit être exécutable.
- Ce programme 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 programme affichera "Good job !", sinon il affchera "Nope, sorry..." - suivis d'un retour à la ligne.
- Si le nombre de paramétres passés au programme 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 XIV

Exercice 11 : count_it

	Exercice : 11
Compter et mesurer des paramètres	
Dossier de rendu : $ex11/$	
Fichiers à rendre : `count_it.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `count_it.rb`.
- Ce programme doit être exécutable.
- Le programme 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 méthode each.

Chapitre XV

Exercice 12 : string_are_arrays

	Exercice : 12
Mieux comprendre les chaînes de caractères	
Dossier de rendu : $ex12/$	
Fichiers à rendre : `string_are_arrays.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `string_are_arrays.rb` qui prend en paramètre une chaîne de caractères.
- Ce programme doit être exécutable.
- Lorsqu'on l'exécute, le programme 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 XVI

Exercice 13 : append_it

	Exercice : 13
Modifier des chaînes de caractères	
Dossier de rendu : $ex13/$	
Fichiers à rendre : `append_it.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `append_it.rb`.
- Ce programme doit être exécutable.
- Le programme va afficher les paramètres passés en argument, un à un, en retirant la dernière lettre de chaque paramètre 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.`

Chapitre XVII

Exercice 14 : free_range

	Exercice : 14
D'ici à là, il n'y a qu'un pas !	
Dossier de rendu : $ex14/$	
Fichiers à rendre : `free_range.rb`	
Fonctions Autorisées : `Toutes`	

- Créez un programme `free_range.rb` qui prend deux paramètres.
- Ces deux paramètres seront deux nombres.
- Ce programme doit être exécutable.
- Vous devez construire un array contenant toutes les valeurs entre ces deux nombres en utilisant un range. Vous afficherez ensuite l'array avec la méthode p.
- Si le nombre de paramètres est différent de 2, vous afficherez `'none'` suivi d'un retour à la ligne.

```
?> ./free_range.rb | cat -e
none$
?> ./free_range.rb 10 14 | cat -e
[10, 11, 12, 13, 14]$
?>
```

`Google range.`