Sujet K3.A :

Votre tâche dans ce kata est d'implémenter une fonction qui calcule la somme des entiers à l'intérieur d'une chaîne de caractères. Par exemple, dans la chaîne

« The30quick20brown10f0x1203jumps914ov3r1349the102l4zy dog »,

la somme des entiers est 3635.

Remarque : seuls les nombres entiers positifs seront testés. Pour rappel, Number(*str*) OU *+str* OU *parseInt(str)* permet de transformer un string *str* en nombre.

Sujet K3.B

Vous devez implémenter une fonction qui renvoie la différence entre la plus grande et la plus petite valeur d'un tableau donné (nommé array) reçu en paramètre. Le tableau peut également contenir des nombres positifs et négatifs.

Si array est vide ou contient un seul élément, la fonction renvoie 0. Aussi, array n'est PAS forcément trié comme sur les exemples.

Exemples :

[1, 2, 3, 4] // renvoie 3 car 4 - 1 == 3

[1, 2, 3, -4] // renvoie 7 car 3 - (-4) == 7

Sujet K3.C

Ami guerrier du code, nous avons besoin de ton aide ! Il semble que nous ayons perdu l'un des éléments de notre séquence, et nous avons besoin de votre aide pour le retrouver !

Notre séquence était censée contenir tous les entiers de 0 à 9 (sans ordre particulier), mais l'un d'entre eux semble avoir disparu.

Écrivez une fonction qui accepte une séquence d'entiers uniques compris entre 0 et 9 (inclus) et qui renvoie l'élément manquant.

Exemples : [0, 5, 1, 3, 2, 9, 7, 6, 4] retourne 8 ; [9, 2, 4, 5, 7, 0, 8, 6, 1] retourne 3

Sujet K3.D

Votre tâche consiste à écrire une fonction appelée *valid_spacing()* ou *validSpacing()* qui vérifie si une chaîne de caractères a un espacement valide. La fonction doit retourner soit true, soit false.

Pour ce kata, la définition d'un espacement valide est un espace entre les mots, sans espace avant ou arrière. Les mots peuvent être n'importe quelle séquence consécutive de caractères sans espace.

Voici quelques exemples de ce que la fonction devrait renvoyer :

* 'Hello world' => true ; * ' Hello world' => false ; * 'Hello world' => false ;

* 'Hello world ' => false ; * 'Hello' => true

Même s'il n'y a pas d'espace, il est toujours valide car il n'y en a pas besoin :

* 'Helloworld' => true ; * 'Helloworld ' => false ; * ' ' => false; * '' => true

Remarque : il n'y aura pas de ponctuation ni de chiffres dans la chaîne d'entrée, mais uniquement des lettres.

Sujet K3.E

Dans cet exercice, une chaîne est transmise à une méthode et une nouvelle chaîne doit être renvoyée avec le premier caractère de chaque mot de la chaîne.

Par exemple :

- « This Is A Test » retourne « TIAT »
- « You earned sugar » retourne « Yes »

Les chaînes de caractères ne contiendront que des lettres et des espaces, avec exactement un espace entre les mots, et aucun espace avant/arrière.

Sujet K3.F

Des personnes ont été tuées ! Vous avez réussi à réduire le nombre de suspects à quelques uns seulement. Heureusement, vous connaissez toutes les personnes que ces suspects ont vues le jour des meurtres.

Vous disposez en guise de paramètres: 1) une liste des noms des personnes décédées : ['Lucas', 'Bill'] par exemple, et 2) d'un dictionnaire contenant tous les noms des suspects et de toutes les personnes qu'ils ont vues ce jour-là, qui peut ressembler à ceci :

```
{
        'James' : ['Jacob', 'Bill', 'Lucas'],
        'Johnny' : ['David', 'Kyle', 'Lucas'],
        'Peter'  : [ 'Lucy', 'Kyle']
}
```

Écrire une fonction qui renvoie le nom du seul tueur, dans notre cas 'James' parce qu'il est la seule personne à avoir vu à la fois 'Lucas' et 'Bill'. On vous garantit qu'il n'y aura toujours qu'un seul tueur. Rappel : for … in… permet de récupérer les clés.

Sujet K3.G

Votre tâche consiste à écrire une fonction qui prend trois entiers a, b et c comme arguments, et qui renvoie True si exactement deux des trois entiers sont des nombres positifs (supérieurs à zéro), et False - dans le cas contraire.

Exemples :

twoArePositive(2, 4, -3) == true

twoArePositive(-4, 6, 8) == true

twoArePositive(4, -6, 9) == true

twoArePositive(-4, 6, 0) == false

twoArePositive(4, 6, 10) == false

twoArePositive(-14, -3, -4) == false

Sujet K3.H

On vous donne un tableau d'entiers de longueur impaire, dans lequel tous les entiers sont identiques, à l'exception d'un seul nombre.

Complétez la fonction qui accepte un tel tableau d’entiers, et qui renvoie ce seul nombre différent.

Le tableau d'entrée sera toujours valide et comportera toujours minimum 3 éléments

Exemples

[1, 1, 2] ==> 2

[17, 17, 3, 17, 17, 17, 17] ==> 3

Sujet K3.I

Écrivez une fonction qui prend une chaîne de caractères et un entier *n* comme paramètres et qui renvoie une liste de tous les mots dont la longueur est supérieure à *n*.

Exemple : l'argument "The quick brown fox jumps over the lazy dog", 4 retourne ['quick', 'brown', 'jumps']

Sujet K3.J

Écrivez une fonction qui combine deux tableaux en prenant alternativement des éléments de chaque tableau.

Exemples :

[‘a’, ‘b’, ‘c’, ‘d’, ‘e’], [1, 2, 3, 4, 5] devient [‘a’, 1, ‘b’, 2, ‘c’, 3, ‘d’, 4, ‘e’, 5]

[1, 2, 3], [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’] devient [1, ‘a’, 2, ‘b’, 3, ‘c’, ‘d’, ‘e’, ‘f’]

Remarques :

- Les tableaux peuvent être de longueurs différentes, avec au moins un caractère/chiffre.
- Un des tableaux contiendra toujours des chaînes de caractères (en minuscules, de 'a' à 'z'), et l'autre contiendra des nombres entiers supérieurs ou égaux à 1.

Sujet K3.K

Étant donné une chaîne de mots séparés par des espaces, retourner le mot le plus long.S'il y a plusieurs mots les plus longs, il renvoie le mot le plus à droite.

Exemples : "red white blue" => "white" ; "red blue gold" => "gold"

Sujet K3.L

Vous aimez le café et souhaitez savoir quels grains vous pouvez acheter en fonction de votre budget.

Votre fonction recevra deux arguments :

1. Un nombre représentant votre budget.
2. Un tableau contenant les prix des différentes variétés de grains de café disponibles.

Votre fonction doit retourner une chaîne contenant les prix des grains de café que vous pouvez vous permettre, triés par ordre croissant, séparés par une virgule.

Exemples :

search(3, [6, 1, 2, 9, 2]) → retourne "1,2,2"

search(14, [7, 3, 23, 9, 14, 20, 7]) → retourne "3,7,7,9,14"

search(0, [6, 1, 2, 9, 2]) → retourne ""