Sujet K1.A :

Nous devons écrire un code pour retourner le prix d'origine d'un produit, le type de retour doit être de type décimal et le nombre doit être arrondi à deux décimales.

On nous donnera le prix de vente (prix réduit) et le pourcentage de vente, notre travail consiste à calculer le prix d'origine. Notre tâche consiste à calculer le prix d'origine :

Pour un article dont le prix de vente est de 75 après application d'une remise de 25 %, la fonction doit renvoyer le prix d'origine de cet article avant application du pourcentage de vente, soit (100.00), bien entendu, arrondi à la deuxième décimale.

Exemple: discoverOriginalPrice(75, 25) => 100.00, où 75 est le prix de vente (prix réduit), 25 est le pourcentage de vente et 100 est le prix d'origine.

Sujet K1.B

Écrire une fonction qui prend un entier num (num >= 0) et insère des tirets ('-') entre chaque deux chiffres impairs de num.

Exemples : 454793 retourne « 4547-9-3 » ; 0 retourne « 0 » ; 1 retourne « 1 » ;

13579 retourne « 1-3-5-7-9 » ; 86420 retourne « 86420 »

Sujet K1.C

Dans ce Kata, votre tâche est de retourner le nombre de paires contenant des nombres consécutifs.

Par exemple : pairs([1,2,5,8,-4,-3,7,6,5]) = 3

Explication: Les paires sont sélectionnées comme suit : [(1,2), (5,8), (-4,-3), (7,6), 5]

- La première paire est (1,2), et les nombres sont consécutifs → Compteur = 1
- La deuxième paire est (5,8), mais ils ne sont pas consécutifs.
- La troisième paire est (-4,-3), consécutifs → Compteur = 2
- La quatrième paire est (7,6), également consécutifs → Compteur = 3
- Le dernier chiffre (5) n'a pas de paire, donc on l'ignore.

Autres tests unitaires :

- expect(pairs([1,2,5,8,-4,-3,7,6,5])).to.equal(3);
- expect(pairs([21, 20, 22, 40, 39, -56, 30, -55, 95, 94])).to.equal(2);
- expect(pairs([81, 44, 80, 26, 12, 27, -34, 37, -35])).to.equal(0);
- expect(pairs([-55, -56, -7, -6, 56, 55, 63, 62])).to.equal(4);
- expect(pairs([73, 72, 8, 9, 73, 72])).to.equal(3);

Sujet K1.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 K1.E

Créez une fonction qui retourne la somme des deux plus petits nombres positifs d'un tableau contenant au minimum 4 entiers positifs.

Aucun nombre à virgule flottante ni entier négatif ne sera présent dans le tableau.

Exemples : sumTwoSmallestNumbers([19, 5, 42, 2, 77]); // retourne 7

sumTwoSmallestNumbers([10, 343445353, 3453445, 3453545353453]); // retourne 3453455

Sujet K1.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 (clé) et de toutes les personnes qu'ils ont vues ce jour-là (valeur), 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 K1.G

C'est le match le plus attendu de l'année scolaire - une compétition entre deux équipes ! Écrivez une fonction qui retourne l'équipe gagnante.

Vous recevrez deux tableaux, chacun contenant deux valeurs :

1. La première valeur est le nombre de points marqués par les poursuiveurs de l'équipe.
2. La deuxième valeur est une chaîne de caractères avec la valeur 'yes' ou 'no' indiquant si l'équipe a capturé un objet spécial (appelé "le vif d'or" dans l'exemple).

L'équipe qui capture ce “vif d’or” (=’yes’) gagne 150 points supplémentaires, mais cela ne garantit pas toujours la victoire.

Exemples :

gameWinners([150, 'yes'],[200, 'no']); // Retourne : ‘team 1 wins!'

gameWinners([400, 'no'],[350, 'yes']); // Retourne : ‘team 2 wins!'

gameWinners([200, 'no'],[50, 'yes']); // Retourne : "draw!"

Remarque : Le jeu ne se termine que lorsqu'une équipe capture l'objet spécial, donc un des tableaux inclura toujours 'yes' ou 'no'. Le score des poursuiveurs peut être n'importe quel entier positif.

Sujet K1.H

Leo veut savoir combien de cadeaux il peut acheter avec son budget. La fonction prend en paramètre son budget et une liste de prix de cadeaux. Elle retourne le nombre maximum de cadeaux que Leo peut acheter.

Exemple : Budget : 20 ; Liste des cadeaux : [13, 2, 4, 6, 1]

Retourne : 4

Sujet K1.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 K1.J

Réorganisez les caractères d'une chaîne selon les positions indiquées dans un tableau d'indices.

Exemple : rearrangeString("abcd", [0, 3, 1, 2]); // Retourne : "acdb"

Explication :

- 'a' va à l'index 0.
- 'b' va à l'index 3.
- ‘c' va à l'index 1.
- 'd' va à l'index 2.

Les longueurs de la chaîne et du tableau sont toujours égales.

Sujet K1.K

Certaines personnes n'ont qu'un prénom, d'autres ont un prénom et un nom de famille, et certaines ont un prénom, plusieurs prénoms secondaires et un nom de famille.

Écrivez une fonction qui initialise les prénoms secondaires, c'est-à-dire que chaque prénom secondaire doit être remplacé par sa première lettre suivie d'un point (.).

Exemples :

InitializeNames('Jack Ryan'); // Retourne : 'Jack Ryan'

initializeNames('Lois Mary Lane'); // Retourne : 'Lois M. Lane'

initializeNames('Dimitri'); // Retourne : 'Dimitri'

initializeNames('Alice Betty Catherine Davis'); // Retourne : 'Alice B. C. Davis'

La fonction doit gérer n'importe quel nombre de prénoms secondaires tout en conservant le prénom et le nom de famille inchangés.

Sujet K1.L

Trouve simplement la valeur la plus proche de zéro dans le tableau. Notez qu'il peut y avoir des valeurs négatives dans le tableau.Le tableau n'est jamais vide et contient uniquement des entiers.

Retourne *null* si il est impossible de déterminer une seule valeur proche de zéro (il y en a plusieurs). Bien sûr, 0 est considéré comme la valeur la plus proche de zéro.

Exemples d’entrées => sorties :

[2, 4, -1, -3] => -1; [5, 2, -2] => null;

[5, 2, 2] => 2; [13, 0, -6] => 0