Lycée Les 3 Sources **Bourg-Lès-Valence**
NSI 1re – Thème 6 Année **2022-23**

Python – Spécification et vérification

Tester des fonctions déjà écrites

Exercice 1 : Le fichier `verification.py` qui se trouve dans `Echange` contient un certain nombre de fonctions. Certaines de ces fonctions ne fonctionnement pas comme prévu. La spécification de chaque fonction est donnée en docstring. Il y a également quelques assertions qui sont toutes validées par les fonctions en l'état. Mais ces tests ne sont pas suffisants. Le code de chaque fonction est doublé pour garder la version d'origine et pour en faire une version corrigée, si nécessaire. Pour chaque fonction :

1) rajouter plus de tests pour déterminer si elle est correcte ou non ;
2) corriger la 2e version de la fonction afin qu'elle passe vos nouveaux tests.

Écrire des fonctions et les tester

Pour chacun des exercices suivants, vous devez écrire le code des fonctions et rajouter des assertions permettant de vérifier la validité de votre fonction.

Exercice 2 : Écrire une fonction `compter(symbole, texte)` qui renvoie le nombre de fois où `symbole` apparaît dans `texte`.

```
>>> compter('b', 'bulle')
1
>>> compter('l', 'bulle')
2
```

Exercice 3 : Écrire une fonction `tous_egaux(liste)` qui renvoie un booléen indiquant si les éléments de la liste sont tous égaux. Dans le cas de la liste vide, on considère que c'est vrai.

```
>>> tous_egaux([1, 1, 1, 1])
True
>>> tous_egaux([1, 2])
False
```

Exercice 4 : Écrire une fonction `derniere_position(val, liste)` qui renvoie la position de la dernière occurrence de `val` dans `liste`. La fonction renvoie `-1` si `val` n'est pas dans `liste`.

```
>>> derniere_position(7, [5, -1, 7, 4, 6, 4, 2])
2
>>> derniere_position(4, [5, -1, 7, 4, 6, 4, 2])
5
>>> derniere_position(0, [5, -1, 7, 4, 6, 4, 2])
-1
```

Exercice 5 : Écrire une fonction `ecart_max_min(liste)` qui renvoie la différence entre le maximum et le minimum de `liste`. On suppose que la liste n'est pas vide.
Vous ne pouvez pas utiliser les fonctions Python **`min`** et **`max`**.
Il est possible de faire cette fonction avec un seul parcours de la liste.

```
>>> ecart_max_min([1, 7]) # mini = 1, maxi = 7 -> 7-1 = 6
6
>>> ecart_max_min([3, 1, 9, 2]) # mini = 1, maxi = 9 -> 9-1 = 8
8
>>> ecart_max_min([15, 1, 9, 2]) # mini = 1, maxi = 15 -> 15-1 = 14
14
```

Exercice 6 : Écrire une fonction `que_les_negatifs(liste)` qui prend une liste `liste` et renvoie une nouvelle liste ne contenant que les éléments strictement negatifs de `liste`.

```
>>> que_les_negatifs([-3, 5, -6, 9])
[-3, -6]
>>> que_les_negatifs([-1, -7, -5, -3])
[-1, -7, -5, -3]
```

Exercice 7 : Écrire une fonction `que_croissant(liste)` qui prend une liste de nombres et renvoie une nouvelle liste commençant par le premier élément de `liste` et en ne rajoute un élément que s'il est strictement supérieur au dernier ajouté.

```
>>> que_croissant([1, 2, 5, 3, 10, -1]) # 3 et -1 inférieur au précédent
[1, 2, 5, 10]
```

Exercice 8 : Écrire une fonction `occurrence_max(liste)` qui renvoie l'élément de la liste non vide `liste` qui apparaît le plus de fois dans la liste. En cas d'égalité, on renvoie n'importe lequel des éléments le nombre maximum de fois. On pourra reprendre la fonction `nb_occurrences` de l'exercice 1.

```
>>> occurrence_max([4, 2, 6, 4, 1, 2, 4])
4
>>> occurence_max(["chien", "chat", "chat"])
"chat"
```

En cas d'égalité, on pourra utiliser une assertion de la forme :

```
assert occurrence_max([1, 2, 1, 2, 3]) in [1, 2] # égalité entre 1 et 2.
```

Exercice 9 : Écrire une fonction `compter_maximum(liste)` qui prend une liste et renvoie le nombre d'occurrences du maximum de `liste`. Si la liste est vide, il faut renvoyer `0`. Vous ne pouvez ni utiliser les fonctions prédéfinies **`max`** et `count`. Il est possible de trouver le résultat en ne faisant qu'un seul parcours de la liste. Il faut bien réfléchir à ce qu'il faut faire lorsqu'on touve un nouveau maximum.

```
>>> compter_maximum([3, 6, 1])
1
>>> compter_maximum([6, 2, 1, 6, 2, 6])
3
```