Corrigé du TP n°4 - Comptage par dictionnaire - Corrigé

Dans tout ce qui suit, l'appellation "tableau" pour une variable Python désignera indifféremment une liste, un tuple ou une chaîne de caractères.

1. Comptage des éléments d'un tableau

Principe de recherche séquentielle de comptage des éléments d'un tableau

- On crée un dictionnaire vide `dico`.
- On parcourt un à un les éléments du tableau et pour chacun des éléments, si la valeur `v` de l'élément n'est pas une clé de dico, on crée une clé `v` pour dico dont la valeur est initialisée à 1 sinon, la valeur de l'élément est une clé de `dico` et on incrémente la valeur correspondante de 1.
- Une fois le parcours du tableau fini, on renvoie `dico`.

Dans ce T.P., on s'efforcera à programmer des fonctions de complexité linéaire (on considerera (à tort) que la complexité de la recherche d'une clé dans un dictionnaire est constante).
On récupèrera le code de la fonction `frazoazar` que l'on avait programmée dans le T.P. n°2.

Q1 : Écrire une fonction `comptage(L)` qui renvoie un dictionnaire dont les clés sont les valeurs des élements de `L` et dont les valeurs associées sont leurs nombres d'occurrences dans `L`.
Exemple de comportement attendu :

```
>>>comptage((3,9,3,1,3,1,3))
{3:4; 9:1, 1:2}
>>>comptage([])
{}
>>>comptage('blabla')
{'b': 2, 'l': 2, 'a': 2}
```

Q2 : Faire le comptage des lettres d'une phrase de 10000 lettres "tirées au hasard" avec la fonction `frazoazar`. Que remarque-t-on et comment l'explique-t-on ?

Q3 : Écrire une fonction `comptage_par2(chaine)` un dictionnaire dont les clés sont les valeurs des mots de 2 caractères successives dans la chaîne de caractères `chaine` (dont on suppose qu'elle a au moins 2 caractères) et dont les valeurs associées sont leurs nombres d'occurrences dans `chaine`.
Exemple de comportement attendu :

```
>>>comptage_par2('blabla')
{'bl': 2, 'la': 2, 'ab': 1}
```

Q4 : Écrire une fonction `comptage_par(k,chaine)` qui donne un résultat analogue à `comptage_par2(chaine)` mais pour les mots de `k` caractères successifs.

Q5 : (Facultatif) Écrire une fonction `comptage_parauplus(k,chaine)` qui donne un résultat analogue à `comptage_par(k,chaine)` mais pour les mots d'**au plus** `k` caractères successifs.

Correction.

Q1 :

```
def comptage(L):
    dico={}
    for e in L:
        if e in dico.keys():
            dico[e]+=1
        else:
            dico[e]=1
    return dico
```

Q2 :

```
from random import *

M=[chr(i) for i in range(97,123)]+[' ']

def frazoazar(n):
    chaine = ''
    l=len(M)
    for k in range(n):
        i = randint(0,l-1)
        chaine += M[i]
    return chaine
```

```
>>>comptage(frazoazar(10000))
{'z': 351, 's': 433, 'h': 404, 'w': 376, 'b': 353, 't': 354, 'k': 360, 'c
    ': 357, 'q': 349, 'r': 357, 'm': 369, 'i': 365, 'y': 356, 'x': 378, '
    u': 348, 'a': 395, 'v': 360, 'l': 361, 'd': 355, 'n': 358, 'e': 420,
    'g': 393, 'j': 380, ' ': 365, 'p': 375, 'f': 360, 'o': 368}
```

On remarque que le nombre d'apparitions de chaque lettre "tourne" autour de $10000/27 \simeq 370$. Cela s'explique grâce à la loi des grands nombres !

Q3 :

```python
def comptage_par2(chaine):
    dico={}
    for i in range(len(chaine)-1):
        e = chaine[i]+chaine[i+1] # ou chaine[i:i+2]
        if e in dico.keys():
            dico[e]+=1
        else:
            dico[e]=1
    return dico
```

Q4 :

```python
def comptage_par(k,chaine):
    dico={}
    for i in range(len(chaine)-k+1):
        e = chaine[i:i+k]
        if e in dico.keys():
            dico[e]+=1
        else:
            dico[e]=1
    return dico
```

Q5 :

```python
def comptage_parauplus(k,chaine):
    dico={}
    for i in range(len(chaine)-k+1):
        for j in range(1,k+1):
            e = chaine[i:i+k]
            if e in dico.keys():
                dico[e]+=1
            else:
                dico[e]=1
    return dico
```