Chapitre 2 : Les dictionnaires

I- Présentation (Y34MWL)

On peut accéder à un élément d'une liste grâce à son indice. Un **dictionnaire** en Python va aussi permettre de rassembler des éléments délimités par des accolades { }, mais ceux-ci seront accessibles par des **clés**. Les clés sont non mutables et uniques alors que les valeurs associées sont des objets quelconques et mutables. Voici quelques opérations utiles :

- Création d'un dictionnaire	`dico={}#dictionnaire vide` `dico={clé1:elt1, clé2:elt2}`
- Ajout d'une valeur	`dico[nouvelle_clé]=nouvelle_valeur`
- Appel d'une valeur	`valeur = dico[clé]`

- Accès aux clés	Avec le mot *in* :
	`liste_cle=[]`
	`for cle in dico:`
	`  liste_cle.append(cle)`
	On peut tester la présence d'une clé
	`cle in dic0 =>(True ou False)`

- Accès aux couples (clé,valeur)	`for cle in dico:` `  print("La clé "+str(cle)+" est associée à l'elt "+str(dico[cle]))`
- Modification d'un élément associé à une clé	`dico[clé]=elt_nouveau` clé est donc une clé déjà déclarée (sinon cela revient à rajouter un élément)
- Longueur d'un dictionnaire	*len(dictionnaire)*

II- Exemple d'utilisation pour le comptage

On souhaite créer une liste permettant de compter le nombre d'occurrences de chaque élément dans une liste L d'entiers.

Par exemple :

```
L=[0,5,10,10,3]
On obtient le comptage suivant :
[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 2]
```

1e méthode :

```
def comptage(L):
    maximum=max(L)
    compteur=[0 for i in range(maximum+1)]
    for i in L:
        compteur[i]=compteur[i]+1
    return compteur
```

Le problème de la méthode précédente est son manque d'efficacité lorsque la dispersion des valeurs de L est importante. Les éléments de L constitueront les clés, la valeur associée dans le dictionnaire est le nombre d'occurrences.

```
def comptage2(L):
    dico={}
    for i in L:
        if i in dico:
            dico[i]=dico[i]+1
        else :
            dico[i]=1
    return dico
```

```
L=[0,5,10,10,3]
On obtient le comptage suivant :
{0: 1, 10: 2, 3: 1, 5: 1}
```

Commenté [AM1]: Séquence de données itérables (on peut avoir accès à ses clés et valeurs par une boucle for) mais non indiçable (l'accès aux éléments ne se fait pas un indice). Un dictionnaire est aussi appelé tableau associatif.
Exemple d'utilisation des listes : lors d'une acquisition en physique d'un signal : l'index est associé au numéro de l'échantillon et est suffisant pour le manipuler
Exemple d'utilisation des dictionnaires : une structure donnant le nom des villes avec le nombre d'habitants.
Il serait possible de faire une liste de listes avec l'exemple précédent, mais la recherche du nombre d'habitants dans une ville ne serait pas aisée.

Commenté [AM2]: Objet de type *dict*
La création d'un dictionnaire peut alors aussi s'écrire :
$dico = dict(cle1 = elt1, cle2 = elt2 \ldots)$

Commenté [AM3]: Une clé ne peut pas être une liste car mutable : la clé peut être de n'importe quel type non mutable : entier, float, chaîne, tuple

Commenté [AM4]: On peut créer un dictionnaire par compréhension :
dico={i : i*10 for i in range(10)}

Commenté [AM5]: Avec le mot *keys* on peut aussi avoir accès aux clés:
`print(dico.keys())`
Avec le mot *values* on peut aussi avoir accès aux clés:
`print(dico.values())`
Avec le mot *items* :
`print(dico.items())`

On peut aussi itérer sur le couple (cle,valeur) :
```
for clé,valeur in dico.items():
    print("la clé",clé,"donne accès à",valeur)
```
...

Commenté [AM6]: On obtient :
`{0: 1, 10: 2, 3: 1, 5: 1}`

Commenté [AM7]: Avec une liste de listes, on peut proposer :
```
def comptage3(L):
  liste=[]
  for i in range(len(L)):
    if L[i] not in L[:i]:
```
...