Python – Dictionnaires

Parcours de dictionnaires

Les dictionnaires se parcourent en Python comme les listes et les tuples.

```
def age_maximum(dico):
    age_max = 0
    plus_vieux = ""
    for nom in dico:
        if dico[nom] > age_max:
            age_max = dico[nom]
            plus_vieux = nom
    print(f"{plus_vieux} est le plus vieux avec {age_max} ans")
```

```
>>> age_maximum({'Alice': 23, 'Bob': 32, 'Charlie': 56, 'Daniel': 10})
Charlie est le plus vieux avec 56 ans
```

Nous allons faire une fonction qui analyse un texte et compte les occurences des différents mots de ce texte. Pour cela nous allons utiliser les deux fonctions suivantes.

```
def occurences(texte):
    dico_occur = dict()
    nouveau = ""
    for s in texte: # On enlève tout ce qui n'est pas une lettre
        if s in ",.;:/!?()«»_-'\"+*1234567890?":
            nouveau += " " # On met un espace à la place
        else:
            nouveau += s
    for mot in nouveau.split():  # On coupe à chaque espace
        mot = mot.lower()
        if mot in dico_occur:
            dico_occur[mot] += 1
        else:
            dico_occur[mot] = 1
    return dico_occur

def occurences_dans_fichier(fichier):
    with open(fichier, 'r', encoding='utf8') as f:
        dico = occurences(f.read())
    return dico

dico = occurences_dans_fichier("LesMiserables.txt")
```

Pour les exercices suivants, nous utiliserons le fichier `LesMiserables.txt` obtenu grâce au site http://www.gutenberg.org.

Exercice 1 : Déterminer le nombre de mots différents dans Les Misérables.

Exercice 2 : Écrire une fonction `mot_le_plus_frequent(dico)` qui renvoie le mot le plus fréquent dans le dictionnaire `dico` associant des mots à leur nombre d'occurences.

Exercice 3 : Écrire une fonction `le_plus_frequent(dico, n)` qui renvoie le mot de `n` lettre le plus fréquent d'après le dictionnaire `dico`. S'il n'y a pas de mot de longueur `n`, renvoyez un texte vide.

Exercice 4 : Écrire une fonction `plus_frequents(dico)` qui affiche les messages suivants, où `MOT` et `NB` sont remplacés par les bonnes valeurs. Vous déterminerez le nombre de lettres maximum à traiter.

```
>>> plus_frequents(dico)
1 lettre(s) : MOT avec NB occurences
2 lettre(s) : MOT avec NB occurences
3 lettre(s) : MOT avec NB occurences
...
```

Utilisation des ensembles

Exercice 5 : Écrire une fonction `inclusion(ens1, ens2)` qui renvoie `True` si et seulement si l'ensemble `ens1` est inclus dans `ens2`.

```
>>> inclusion(set(), {1, 2, 3})
True
>>> inclusion({1, 3}, {1, 2, 3})
True
>>> inclusion({1, 5}, {1, 2, 3})
False
```

Exercice 6 : Écrire une fonction `union(ens1, ens2)` qui renvoie un nouvel ensemble égal à l'union des ensembles `ens1` et `ens2`.

```
>>> union({1, 4}, {2, 4, 7})
{1, 2, 4, 7}
```

Exercice 7 : Écrire une fonction `intersection(ens1, ens2)` qui renvoie un nouvel ensemble égal à l'intersection des ensembles `ens1` et `ens2`.

```
>>> intersection({7, 1}, {2, 9})
set()
>>> intersection({7, 1, 6}, {2, 6, 11})
{6}
```

Exercice 8 : Écrire une fonction `soustraction(ens1, ens2)` qui renvoie un nouvel ensemble contenant les éléments de `ens1` qui ne sont pas dans `ens2`.

```
>>> soustraction({6, 7}, {1, 7, 6})
set()
>>> soustraction({12, 6, 7, 1}, {1, 7, 9})
{12, 6}
```

Exercice 9 : On prend un nombre n et on additionne le carré de ses chiffres. On recommence la même chose à partir du résultat obtenu. On répète le processus jusqu'à obtenir soit 1, soit un nombre déjà obtenu. Si on obtient 1, on dit que le nombre n est **heureux**. Par exemple, 23 est heureux :

$$23 \to 2^2 + 3^2 = 13 \to 1^2 + 3^2 = 10 \to 1^2 + 0^2 = 1$$

Èrire une fonction `heureux(n)` qui prend un entier positif `n` et renvoie un booléen indiquant s'il est heureux ou pas. Afin de tester si le nombre a déjà été obtenu au cours du processus, vous pouvez utiliser un ensemble.