TP 3

Dictionnaires

I - Dictionnaires

I.1 - Définitions

Une liste est une structure permettant de stocker des éléments et d'y accéder à travers leur indice c'est-à-dire leur position dans la liste.
Un dictionnaire est une structure qui permet également de stocker des éléments mais de manière différente, les indices de la liste étant remplacés par des clés.
Ainsi, un dictionnaire est une structure qui associe des valeurs (de type quelconque) à des clés (de type quelconque).

I.2 - Création d'un dictionnaire

- Un dictionnaire **d** peut être construit en listant entre accolades les clés `c_i` et les valeurs associés `v_i` en respectant la syntaxe suivante.

```
d = { c1 : v1 , c2 : v2 , ... , cn : vn }
d = {} # dictionnaire vide
```

Par exemple, le dictionnaire suivant

```
Dico = {"nom" : "Leonard", "jour" : 14, "mois" : 4,  "annee" : 1452}
```

contient 4 clés qui sont ici les chaînes de caractère "nom", "jour", "mois" et "annee", auxquelles sont respectivement associées les valeurs "Leonard", 15, 4 et 1452.

- Un dictionnaire peut être également construit à partir d'un dictionnaire vide auquel on ajoute au fur et à mesure une nouvelle clé et la valeur associée en respectant la syntaxe suivante, l'ordre d'insertion n'ayant aucune importance.

```
d = {} # dictionnaire vide
d[cle1] = valeur1
d[cle2] = valeur2
...
```

Par exemple, on peut aussi créer le dictionnaire Dico construit plus haut de la manière suivante

```
Dico = {}
Dico["nom"] = "Leonard"
Dico["jour"], Dico["mois"], Dico["annee"] = 14, 4, 1452
```

- On peut construire un dictionnaire par compréhension. Par exemple :

```
>>> { x*x : x for x in range(4)}
{0: 0, 1: 1, 4: 2, 9: 3}
```


I.3 - Accès et modifications

Si **d** est un dictionnaire et **c** est une clé, alors :

- L'expression **c in d** renvoie un booléen indiquant si la clé c est présente dans le dictionnaire.
- L'expression **d[c]** renvoie la valeur associée à la clé si celle-ci est présente dans le dictionnaire.
- L'instruction **d[c] = v** crée une nouvelle association si la clé n'est pas présente dans le dictionnaire, et modifie l'association précédente sinon.
- L'instruction **del d[c]** supprime l'entrée associée à la clé c dans le dictionnaire.
- On peut enfin connaitre le nombre de clés présentes dans un dictionnaire avec la fonction **len(d)**.

Par exemple

```
Dico["nom"] = "Leonard De Vinci"    # Modification
Dico["pays"] = "Italie"    # Création d'une nouvelle clé/valeur.
Dico["annee"] += 1   # modification de l'année (+1)
Dico["annee"]    # accède à la valeur associée à la clé "annee"
len(Dico)   # renvoie 5 (la longueur du dictionnaire)
del Dico["pays"]
"pays" in Dico   # renvoie False (on vient de le supprimer)
```

I.4 - Parcours d'un dictionnaire

- On peut parcourir toutes les clés d'un dictionnaire **d** avec la boucle **for** (l'ordre de parcours est arbitraire).

```
for c in Dico :
    print("la clé", c, "est associée à la valeur", d[c])
```

- On peut obtenir une liste contenant toutes les valeurs du dictionnaire **d** avec l'instruction **d.keys()** ou plus simplement avec **list(d.keys())**

```
>>> d = {"a" : 1 , "b" : 2 , "c" : 1}
>>> list(d.keys())
['b', 'a', 'c']
```

- On peut obtenir une liste contenant toutes les valeurs avec **list(d.values()**

```
>>> list(d.values())
[1, 1, 2]
```

- On peut obtenir un tableau contenant toutes les entrées du dictionnaire sous la forme (clé,valeur) avec **list(d.items())**

```
>>> list(d.items())
[('a', 1), ('b', 2), ('c', 1)]
```

II - Exercices

Exercice 1

Au scrabble, on marque des points en faisant la somme des points attribués à chaque lettre d'un mot.
Les points répartis sont les suivants :

- 1 point pour les lettres A,E,I,L,N,O,R,S,T et U ;
- 2 points pour D,G et M ;
- 3 points pour B, C et P ;
- 4 points pour F, H et V ;
- 8 points pour J et Q ;
- 10 points pour K, W, X, Y et Z.

1. Définir un dictionnaire `DicoValeurs` dont les clés sont les lettres de l'alphabet et les valeurs leur score.
2. Ecrire une fonction `score(DicoValeurs,mot)` qui prend en entrée le dictionnaire `DicoValeurs` et un mot écrit sous forme d'une chaine de caractères composée de lettres majuscules non accentuées et qui renvoie le score associé au mot.
3. Vérifier que le score du mot `"BRAVO"` vaut bien 10.

Exercice 2

1. Écrire une fonction `occurences(L)` prenant en entrée une liste `L` et renvoyant un dictionnaire dont les clés sont les éléments de `L` et les valeurs le nombre de fois où chaque élément apparaît dans `L`.
2. Une première application.
   (a) Écrire une fonction `sans_doublon(L)` prenant en entrée une liste `L` et renvoyant une autre liste contenant les mêmes éléments, mais une seule fois.
   (b) Écrire une fonction `doublons(L)` prenant en entrée une liste `L` et renvoyant une autre liste, contenant tous les éléments de `L` qui apparaîssent au moins deux fois
3. Une autre application.
   Quel est le mot de 7 lettres qui apparaît le plus souvent dans le texte "Le tour du monde en quatre-vingt jours" de Jules Verne?
   Pour répondre à cette question, on va construire une liste contenant tous les mots du texte puis créer le dictionnaire des occurences de ces mots.
   (a) Exécuter les instructions suivantes qui permettent de créer une liste nommée `texte` contenant tous les mots du texte de Jule Verne.

```
f = open("ADRESSE/ltdme80j.txt") # ouverture du fichier
texte = f.read().split() # lecture du fichier et stockage des mots
f.close()  # fermeture du fichier
```

   (b) Créer à partir de la liste `texte` le dictionnaire des occurences des mots du texte.
   (c) On suppose que `d` est un dictionnaire dont les clés sont les mots du texte et les valeurs les occurences de ces mots.
   Écrire une fonction `plus_frequent(d, k)` qui renvoie le mot de `k` lettres qui est associé au dictionnaire `d` avec la plus grande valeur.
   En cas d'égalité, on choisira arbitrairement et s'il n'y a aucun mot de `k` lettres dans `d`, on renverra une chaîne vide.
   (d) Répondre à la question.