Correction du TP 8

Exercice 1 — Maison, tu n'as pas de clé

La fonction suivante détermine si un objet apparait comme valeur d'un dictionnaire.

```
def recherche_valeur(Dico, v):
    """
    Entrées : un dictionnaire Dico et un objet v.
    Sortie : True si v est l'une des valeurs du dictionnaire Dico,
             False sinon.
    """
    for cle in Dico: # La variable cle parcourt une à une
                     # les clés du dictionnaire Dico.
        if Dico[cle] == v:
            return True
            # La fonction s'arrête dès que la valeur v est trouvée.
    return False
    # Si à la fin du parcours, la fonction ne s'est pas arrêtée,
    # alors l'objet v n'est pas une valeur du dictionnaire.
```

Exercice 2 — Fréquences et nombres de caractères distincts

1. À partir d'une chaine de caractères, on souhaite créer un dictionnaire dont les clés sont les caractères distincts de la chaine et les valeurs associées les nombres d'occurrences des caractères dans la chaine.

   - **Première méthode : un seul parcours de chaine, avec recherches dans le dico**

```
def frequences(ch):
    """
    Entrée : une chaine de caractères ch.
    Sortie : le dictionnaire dont
        - les clés sont les caractères (distincts) de ch,
        - les valeurs associées sont le nombre d'occurrences
          du caractère dans ch.
    """
    Dico = {}
    for lettre in chaine:
    # lettre parcourt un à un les caractères de la chaine ch.
        if lettre in Dico :
        # Si la lettre est une clé de Dico,
        # c'est-à-dire si le caractère lettre a déjà été rencontré,
        # alors on augmente de 1 son nombre d'occurrences.
            Dico[lettre] += 1
        else:
            Dico[lettre] = 1
    return Dico
```

- **Seconde méthode : deux parcours de chaine, sans recherche dans le dico**
  Une autre méthode consiste à faire deux parcours de la chaine.
  Dans un premier temps, on crée une entrée dans le dictionnaire dont la clé est le caractère et la valeur associée est nulle. Si un même caractère apparait plusieurs fois, l'entrée précédente du dictionnaire portant la même clé sera écrasée. Donc à la fin du parcours, les clés du dictionnaire seront exactement tous les caractères distincts de la chaine.
  Dans un second temps, on parcourt à nouveau la chaine et on incrémente de 1 le nombre d'occurrences de chaque caractère rencontré, c'est-à-dire la valeur associée à la clé correspondante dans le dictionnaire.

```
def frequences_bis(ch):
    Dico = {}
    for lettre in ch:
        Dico[lettre] = 0
    # Chaque caractère de la chaine est maintenant
    # une clé du dictionnaire.
    # On compte alors les occurrences de chaque caractère.
    for lettre in ch:
        Dico[lettre] += 1
    return Dico
```

  La création du dictionnaire peut aussi se faire en compréhension.

```
def frequences_ter(ch):
    Dico = {lettre : 0 for lettre in ch}
    for lettre in ch:
        Dico[lettre] += 1
    return Dico
```

2. Le nombre de caractères distincts de la chaine est le nombre de clés du dictionnaire précédent, c'est-à-dire sa longueur.

```
def nombre_lettres_distinctes(ch):
    """
    Entrée : une chaine de caractères ch.
    Sortie : le nombre de caractères distincts de la chaine ch.
    """
    return len(frequences(ch))
```

Exercice 3 — Maximum d'un dictionnaire

1. La fonction suivante renvoie le maximum des valeurs d'un dictionnaire donné en argument.

```
def maximum(Dico):
    """
    Entrée : un dictionnaire Dico à valeurs numériques.
    Sortie : la valeur maximale présente dans Dico.
    """
    Liste_cles = list(Dico.keys()) # Liste de toutes les clés de Dico.
    une_cle = Liste_cles[0] # Une des clés du dictionnaire.
    valeur_max = Dico[une_cle] # Une valeur du dictionnaire,
```

```
                                # pour initialiser.
    for cle in Dico: # cle parcourt toutes les clés du dictionnaire.
        if Dico[cle] > valeur_max:
            valeur_max = Dico[cle]
    return valeur_max
```

2. La fonction suivante renvoie une clé dont la valeur associée est le maximum des valeurs du dictionnaire passé en argument.

```
def cle_du_maximum(Dico):
    """
    Entrée : un dictionnaire Dico à valeurs numériques.
    Sortie : une clé dont la valeur associée dans Dico
             est la valeur maximale.
    """
    Liste_cles = list(Dico.keys())
    cle_du_max = Liste_cles[0]
    # On stockera une clé dont la valeur associée
    # est la plus grande valeur rencontrée.
    for cle in Dico:
        if Dico[cle] > Dico[cle_du_max]:
            cle_du_max = cle
    return cle_du_max
```

Exercice 4 — Index

1. Pour construire un dictionnaire dont les clés sont les éléments distincts d'une liste et les valeurs associées les listes de tous les indices des occurrences de l'élément dans la liste, on commence par créer un dictionnaire dont les valeurs sont toutes la liste vide, sur le même principe que dans l'exercice 2.

   On parcourt ensuite les indices de la liste : pour chaque indice k de la liste L, la valeur associée dans le dictionnaire à la clé $L[k]$, qui est une liste, est allongée de l'élément k.

```
def occurrences(L):
    """
    Entrée : une liste L.
    Sortie : le dictionnaire dont
        - les clés sont les éléments (distincts) de L,
        - les valeurs associées sont la liste des indices
          des occurrences de l'élément dans la liste L.
    """
    Dico = {element : [] for element in L}
    # Les clés du dictionnaire Dico sont maintenant
    # les éléments distincts de la liste L.
    for k in range(len(L)): # k va de 0 à len(L) - 1,
    # c'est-à-dire que k parcourt les indices des éléments de L.
        Dico[L[k]].append(k)
        # On ajoute l'indice k comme nouvel élément
        # de la liste Dico[L[k]], c'est-à-dire de la valeur
        # associée à la clé L[k] dans le dictionnaire.
    return Dico
```

2. Pour obtenir le nombre d'occurrences de chaque élément de la liste à partir du dictionnaire donné par la question précédente, il suffit pour chaque élément de la liste, c'est-à-dire pour chaque clé du dictionnaire, de calculer la longueur de la liste de ses occurrences, c'est-à-dire la longueur de la valeur associée à la clé dans le dictionnaire.

```
def frequences(L):
    """
    Entrée : une liste L.
    Sortie : le dictionnaire dont
        - les clés sont les éléments (distincts) de L,
        - les valeurs associées sont le nombre d'occurrences
          de l'élément dans la liste L.
    """
    Dicoccurrences = occurrences(L)
    return {cle : len(Dicoccurrences[cle]) for cle in Dicoccurrences}
```

3. On construit la liste des éléments n'apparaissant qu'une seule fois dans la liste à partir du dictionnaire donné par la question précédente.

```
def uniques(L):
    """
    Entrée : une liste L.
    Sortie : la liste des éléments de L qui apparaissent
             une seule fois dans la liste L.
    """
    Dicofrequences = frequences(L)
    Uniques = []
    for cle in Dicofrequences: # cle parcourt les clés du dictionnaire,
    # c'est-à-dire les éléments (distincts) de la liste L.
        if Dicofrequences[cle] == 1:
            Uniques.append(cle)
    return Uniques
```