Séance 8 : Ensembles, dictionnaires et fichiers texte

L1 – Université Côte d'Azur

Exercice 1 — L'ensemble des lettres minuscules

1. Définissez l'ensemble `minuscules` contenant les 26 lettres minuscules de l'alphabet.
2. Définissez la fonction `nb_occurrences_minuscules(s)` qui prend en argument une chaîne de caractères `s` et qui renvoie le nombre d'occurences d'une lettre minuscule dans `s`, en testant l'appartenance de chaque caractère de `s` à l'ensemble `minuscules`. Par exemple, `nb_occurrences_minuscules('AaAaBz')` renvoie 3.
3. Définissez la fonction `ensemble_minuscules(s)` qui renvoie l'ensemble des lettres minuscules apparaissant dans `s`. Par exemple, `ensemble_minuscules('AaAaBz')` renvoie `{'a','z'}`.
4. Déduisez-en la fonction `nb_minuscules(s)` qui renvoie le nombre de lettres minuscules apparaissant dans `s`.

Exercice 2 — Le temps qui passe

Définissez une fonction `jour(s)` qui crée un fichier `s` dans lequel elle écrit ligne après ligne les différentes heures de la journée, à intervalle de 5 minutes. Le fichier ressemblera à :

```
00:00
00:05
...
23:50
23:55
```

Indication : si `x=42`, la chaîne `f'{x:*>10}'` vaut `'********42'`, c'est-à-dire une chaîne de 10 caractères représentant le nombre `x` aligné à droite en utilisant le caractère `*` comme caractère de remplissage.

Exercice 3 — Liste de contacts

On dispose d'une variable `contacts` contenant une liste de contacts d'un smartphone. Cette liste de contacts est stockée à l'aide d'un dictionnaire Python. Par exemple, on pourrait avoir :

```
>>> print(contacts)
{'Chloé': '0601020304', 'Quentin': '0710203040', 'Lyes': '0623344556'}
```

1. Quelle instruction permet de remplacer le numéro de Chloé par `'0611223344'` ?
2. Quelle instruction permet d'ajouter Sarah dans la liste de contacts, dont le numéro est `'0145444342'` ?
3. Quelle instruction permet d'afficher le numéro de Lyes ?
4. Quelle instruction permet d'effacer Chloé du répertoire ?
5. Écrire une fonction `affichage_détail(contacts)` qui affiche un contact par ligne suivit de son numéro.

```
>>> afficher_contacts(contacts)
Chloé : 0601020304
Quentin : 0710203040
Lyes : 0623344556
```

Exercice 4 — Liste de contacts inversée

1. Écrivez une fonction `inverse_liste_contacts(contacts)` qui prend en argument une liste de contacts sous forme de dictionnaire au même format que l'exercice précédent et qui renvoie le dictionnaire inverse indexé par les numéros au lieu des noms. Par exemple :

```
>>> inverse_liste_contacts(contacts)
{'0601020304': 'Chloé', '0710203040': 'Quentin', '0623344556': 'Lyes'}
```

```python
def inverse_liste_contacts(contacts) :
    res = dict()
    for nom in contacts :
        tél = contacts[nom] # ici nom est la clé et tel la valeur
        res[tél] = nom      # ici tél est la clé et nom la valeur
    return res
```

2. Écrivez une fonction `affiche_liste_appels(L ,contacts)` qui prend en arguments une liste de couples de chaînes de caractères de la forme `(date,numero)` et une liste de contacts, et qui affiche la liste des appels reçus en indiquant si possible le nom de la personne qui a appellé.

   Par exemple, on aura en prenant `L = [ ('10:03', '0623344556'), ('9:45', '0623344556'), ('hier', '0800123123'), ('20/11', '0623344556')]` :

```
>>> affiche_liste_appels(L , contacts2)
10:03 Quentin
9:45 Quentin
hier 0800123123
20/11 Chloé
```

```python
def affiche_liste_appels(L , contacts) :
    contacts_inv = inverse_liste_contacts2(contacts)
    for e in L :
        (heure,tel)=e
        if tel in contacts_inv :
            nom = contacts_inv[tel]
            print(heure, nom)
        else :
            print(heure ,tel)
```

Exercice 5 — Export et import

1. Écrire une fonction `sauvegarde(contacts, chemin)` qui stocke le dictionnaire de contact dans un fichier texte `chemin`. Chaque ligne sera au format `nom,numéro`.

```
Chloé,0601020304
Quentin,0710203040
Lyes,0623344556
```

```python
def sauvegarde(contacts, chemin):
    fichier = open(chemin, 'w', encoding="utf-8")
    for nom in contacts:
        numéro = contacts[nom]
        fichier.write(f"{nom},{numéro}\n")
    fichier.close()
```

2. Écrire une fonction `découper(chaîne)` qui à partir d'une chaîne de la forme `"nom,numéro\n"` renvoie la liste `["nom", "numéro"]`. On pourra utiliser la méthode `split` vue en cours.

```python
def découper(chaîne):
    liste = chaîne.split(",")
    assert len(liste) == 2
    numéro = liste[1]
    if numéro[-1]=="\n":
        numéro = numéro[:-1] # jusqu'au dernier caractère, exclu
    return [ liste[0], numéro ]
```

3. Écrire une fonction `importer(chemin)` qui renvoie le dictionnaire correspondant au chemin.

```python
def importer(chemin):
    fichier = open(chemin, 'r', encoding="utf-8")
    dico = dict()
    for ligne in fichier:
        [nom, numéro] = découper(ligne)
        dico[nom] = numéro
    fichier.close()
    return dico
```

Exercice 6 — Liste de contacts inversée avec répétitions

On revient sur la fonction `inverse_liste_contacts` de l'exercice ci-dessus.

1. Que renvoie votre fonction pour la liste de contacts suivante ?

```python
contacts2 = {'Maison Chloé': '0901020304', 'Maison Lyes': '0901020304',
             'Alex': '0412345678'}
```

```
>>> inverse_liste_contacts(contacts2) # On a perdu Chloé !
{'0901020304': 'Maison Lyes', '0412345678': 'Alex'}
```

2. Écrivez la fonction `inverse_liste_contacts2(contacts)` qui prend en argument une liste de contacts comme dans l'exercice précédent et renvoie la liste indexée par les numéros au lieu des noms en listant les noms de toutes les personnes à qui appartient le numéro. Par exemple :

```
>>> inverse_liste_contacts2(contacts2)
{'0901020304': ['Maison Chloé', 'Maison Lyes'], '0412345678': ['Alex']}
```

```python
def inverse_liste_contacts2(contacts) :
    res = dict()
    for nom in contacts :
        tél = contacts[nom]
        if res.get(tél,None)==None:
            res[tél] = [nom]
        else:
            res[tél].append(nom)
    return res
```

Exercice 7 — Le jeu de Nim

Le jeu de Nim que l'on considère fait s'affronter deux joueurs qui doivent à tour de rôle retirer des allumettes d'un tas contenant initialement n allumettes. À achaque tour, un joueur peut retirer 1, 2 ou 3 allumettes. Le joueur qui retire la dernière allumette a perdu.

Par exemple, en partant de 5 allumettes, on a la partie 5,2,1,0 perdue par le joueur qui commence. Dans ce cas précis, le joueur qui joue en second a même une stratégie gagnante : quel que soit le nombre d'allumettes que le joueur qui commence retire au premier tour, il peut se débrouiller pour laisser une seule allumette à la fin du second tour.

1. Écrivez une fonction `nim_gagnant(n)` qui renvoie `True` si le joueur qui commence a une stratégie gagnante et `False` sinon. Par exemple, `nim_gagnant(5)` renvoie `False` mais `nim_gagnant(4)` renvoie `True`.

```python
def nim_gagnant(n):
    if n<=0:
        return True
    for i in [1,2,3]:
        if not nim_gagnant(n-i):
            return True

    return False
```

2. Écrivez une fonction `nim_memo(n)` qui renvoie les mêmes résultats en utilisant la mémoïsation.

```
mem=dict()

def nim_memo(n):
    if n<=0:
        return True
    if n in mem:
        return mem[n]
    for i in [1,2,3]:
        if not nim_memo(n-i):
            mem[n]=True
            return True
    mem[n]=False
    return False
```

3. Comparez le temps de calcul pour `n=33`.

```
>>> d1 = chronomètre(nim_gagnant,33)
>>> d2 = chronomètre(nim_memo,33)
>>> d1/d2 # La différence est impressionnante (un facteur d'environ 200 000)
147041.74603174604
```