1 Compréhension du cours

1.1 Lecture de code

Indiquer si les codes suivants contiennent ou déclenchent des erreurs à l'exécution et préciser lesquelles.

a.
```
with open('dates.txt', 'r') as f:
    i=0
    while i < len(f):
        afficher_date(f[i])
        i = i + 1
```

b.
```
f = open(dates.txt, 'r'):
    for date in f:
        afficher_date(date)
```

c.
```
with open('dates.txt', 'r') as f:
    lignes = f.readlines()
    f.write('25;12;30;\n')
```

d.
```
# l est de type list[str]
with open('positions.txt', 'w') as f:
    f.writelines(l[:-1])
f.write('\t'+l[-1] +'\n')
```

1.2 Manipulations de fichiers

On considère un fichier `liste_courses.txt` contenant les lignes suivantes (une liste de courses) et dont la dernière ligne est terminée par un retour chariot :

```
lentilles
pommes
carottes
```

On considère le code suivant, qui a l'intention de rajouter du pain et du chocolat sur la liste :

```
>>> with open('liste_courses.txt', 'r') as entree:
    # liste = readlines(entree) # ???
    liste = entree.readlines()
>>> liste
...
>>> liste.append('chocolat\n')
>>> liste.append('pain\n')
>>> with open('liste_courses.txt', 'w') as entree:
    entree.writelines(liste)
```

1. La ligne commentée utilisant `readlines(entree)` est-elle correcte ?
2. Compléter la valeur de `liste` sur les `...`, puis donner le contenu du fichier `liste_courses.txt`.
3. Si on avait oublié les `\n` dans les appels à `append`, quel aurait été le contenu du fichier ?
4. On souhaite remplacer le mode `w` par le mode `a` dans l'ouverture du fichier. Comment écrire le code pour obtenir la liste de courses de la question 2 ?

1.3 Ouvertures

Le but de l'exercice est d'identifier quel mode d'ouverture utiliser pour un fichier selon la spécification du programme.

On considère un jeu monojoueur qui manipule une grille représentée par une liste de listes et calcule un score en fin de partie gagnante. Les joueurs utilisent le surnom unique qui est associé à leur compte sur la plateforme de jeu. Les fonctionnalités proposées incluent :

a. Pour afficher des podiums : enregistrer dans un fichier le score d'un joueur en fin de partie gagnante, à raison d'un fichier par joueur et d'un score par ligne écrit à la fin du jeu en fin de fichier ; le nom de ce fichier est `<surnom>.score`.

b. Pour pouvoir interrompre le jeu quand on le souhaite sans perdre le début de la partie : sauvegarder le contenu de la grille dans un fichier `partie.txt`. On ne peut sauvegarder qu'une seule partie en cours. On efface la partie sauvegardée précédemment si elle existe.

c. Pour pouvoir reprendre une partie interrompue : construire une liste de listes représentant la grille courante qui est décrite dans le fichier `partie.txt`.

5. Pour chacune des fonctionnalités précédentes, décrire l'appel à `open` avec le mode d'ouverture le plus adapté : `'r'`, `'a'` ou `'w'`.

2 Exercices de programmation

2.1 Vide ou pas vide ?

6. Écrire un prédicat `toutes_lignes_non_vides` qui prend en paramètre un nom de fichier existant et renvoie `True` ssi toutes les lignes du fichier sont non vides.

7. Écrire un prédicat `existe_ligne_commencant_par` qui prend en paramètre un nom de fichier existant et une chaîne de caractère non vide, et renvoie `True` ss'il existe dans le fichier une ligne commençant par la chaîne.

2.2 Fusion

On considère des fichiers tels que :

- chaque ligne contient un entier (pour simplifier)
- le fichier est trié par valeur croissante des entiers

On souhaite fusionner 2 tels fichiers, c'est à dire écrire un troisième fichier qui réunit les entiers contenus dans les 2 fichiers, dans l'ordre croissant. On ne se préoccupera pas des éventuels doublons. Par exemple, si le premier fichier contient les entiers `1 4 6 12` et le deuxième fichier contient les entiers `1 2 3 6 20 22` on trouvera dans le troisième fichier les entiers `1 1 2 3 4 6 6 12 20 22`.

8. Proposer une solution qui lit le contenu des 2 fichiers puis passe par un calcul sur une structure de données intermédiaire avant d'écrire le résultat dans un troisième fichier.

9. Proposer une solution qui ne passe pas par une structure de données intermédiaire et écrit le contenu du troisième fichier au fur et à mesure qu'il lit les 2 fichiers.

2.3 Précipitations

On considère un fichier de données `precipitations.csv` qui contient des dates, des mesures de précipitations (on mesure la pluie qui est tombée en millimètres) et de mesure de vent (en km par heure). Par exemple :

```
date;precipitations;vent
20231120;1.4;45.0
20231121;1.4;60.1
20231121;1.4;5.2
20231122;1.2;10.
20231123;0;2.
```

10. Proposer une solution qui permet :

- de calculer la moyenne des précipitations contenues dans le fichier
- de vérifier si oui ou non les dates sont triées par ordre croissant
- de vérifier si oui ou non les dates sont toutes différentes
- d'ajouter une ligne contenant une date et deux mesures (précipitations et vent) au fichier