TD sur la lecture et l'écriture dans un fichier

Dans ce TD, on n'utilise pas l'interface graphique tkinter. Les affichages se font dans le terminal.

Exercice 1: ajouter des moyennes

Récupérer le fichier `notes` qui contient des lignes de note sous le format suivant:

- en premier apparaît le nom de l'étudiant
- ensuite 3 notes sur la même ligne

Il faut écrire un programme qui ajoute à la fin de chaque ligne la moyenne des 3 notes. Le résultat doit être écrit dans un fichier `moyenne`.

Par exemple, si la ligne dans le fichier de notes est

```
Toto 4 11 9
```

alors la ligne correspondante dans le fichier de moyenne doit être

```
Toto 4 11 9 8.0
```

Exercice 2: jouer avec les mots

Récupérer le fichier `words.txt` qui contient un ensemble de mots anglais triés par ordre alphabétique (source ici).

1. Ecrire la fonction

```
def nb_lignes(nom_fichier)
```

qui retourne le nombre de lignes du fichier dont le nom est passé en argument comme chaîne de caractère. Combien de mots contient la liste fournie?

2. Ecrire la fonction

```
def ecrit_liste_mots(n)
```

qui écrit tous les mots de n lettres du fichier `words.txt` dans le fichier `wordsn.txt` (le `n` est à remplacer par sa valeur). Combien y a-t-il de mots de 5 lettres?

3. Ecrire la fonction

```
def melange_mots(fichier)
```

qui mélange tous les mots du fichier dont le nom est passé en argument en écrivant le résultat dans le fichier portant le même nom avec l'extension `.mel`. Pour cela, vous pouvez utiliser les fonction suivantes:

- `fichier.readlines()` qui retourne une liste dont chaque élément est une ligne du fichier lu;
- `random.shuffle(liste)` de la librairie `random` qui mélange les éléments de la liste passée en argument; la liste est directement modifiée par la fonction.

4. Dans le jeu wordle, il faut deviner un mot de 5 lettres en proposant des mots de la même taille: à chaque tentative, on connaît les lettres bien placées et les lettres mal placées c'est-à-dire celles qui sont dans le mot recherché mais situées à un autre endroit dans le mot. Vous pouvez tester le jeu sur ce site en anglais. La vidéo suivante explique comment résoudre le jeu avec un programme informatique, et ce que ça veut dire. Ici nous allons utiliser une méthode qui n'est pas optimale mais qui donne des résultats satisfaisants.

Ecrire la fonction

```
def compare_mots(m1, m2)
```

qui prend en argument deux mots de même taille n et retourne une liste de taille n telle que le i-ème élément vaut

- 1 si la lettre à la position i est identique pour $m1$ et $m2$;
- sinon 2 si la lettre qui est dans $m1$ apparaît dans $m2$ à une position qui n'a pas encore été utilisée pour attribuer la valeur 1 ou 2;
- 0 sinon. Une telle liste est appelé un *profil*. C'est cette information qui est donnée par le jeu quand l'on teste un mot $m1$ et que le mot cherché est $m2$.

Tester votre fonction.

5. Ecrire la fonction

```
def ecrit_liste_compatible(fichier, m, profil)
```

qui prend en entrée un fichier de mots, un mot et un profil et qui écrit dans le fichier qui se nomme `fichier.comp` la liste des mots `m2` tels que l'appel à la fonction `compare_mots(m, m2)` retourne le profil passé en argument.

6. Utiliser ces fonctions pour résoudre le jeu: itérativement, choisir un mot au hasard dans la liste de mots possibles et le tester sur le site du jeu. Récupérer ensuite le profil donné par le jeu, et calculer la liste des mots qui sont compatibles avec ce profil.

7. Proposer une alternative au choix aléatoire du mot.