Exercices de révision : lecture/écriture dans un fichier

Quelques rappels :

- `f=open(nom_du_fichier, mode)` pour ouvrir un fichier. `nom_du_fichier` est une chaîne de caractères contenant le nom du fichier à ouvrir. `mode` est une chaîne de caractères pouvant prendre les valeurs :
  - `'r'`, comme *read* : fichier ouvert en lecture ;
  - `'w'`, comme *write* : fichier ouvert en écriture, le fichier est créé à l'occasion, s'il existait il est écrasé (le contenu précédent est perdu).
  - `'a'`, comme *append* : de même que précédemment, mais si le fichier existe déja, l'écriture se fera à la suite du fichier.

  `f` est ce qu'on appelle un *wrapper*, l'objet qui permet de récupérer des informations (en mode lecture) ou écrire dans le fichier (en mode écriture). Dans ce qui suit, `f` désigne un tel wrapper.
- Parcourir un fichier ouvert en lecture (mode `'r'`), voici deux solutions :
  - une boucle `for` : on peut itérer sur un fichier, `for x in f` fait prendre à la variable `x` successivement comme valeurs les lignes du fichier. Ce sont des chaînes de caractères, terminées par le saut de ligne `'\n'`, comptant comme un seul caractère.
  - `f.readlines()` s'évalue en une liste, dont les éléments sont les lignes du fichier, décrites précédemment.
- Pour écrire dans un fichier (ouvert en écriture), on utilisera `f.write(s)` où `s` est la chaîne de caractères à écrire.
- `f.close()` pour fermer un wrapper.
- Opérations utiles sur les chaînes de caractères. On rappelle que les chaînes de caractères sont immuables. Les méthodes suivantes renvoient de nouveaux objets.
  - `int`, `float`, `str`... sont les fonctions de conversion. Par exemple, `str(4)` s'évalue en `"4"` et `float("2.5")` en `2.5` (type flottant).
  - `+` est la concaténation de chaînes : `"abc" + "def" +"h"` s'évalue en `"abcdefgh"`.
  - `s.split(c)` permet de séparer la chaîne `s` autour du caractère `c`, cette expression s'évalue en une liste. Par exemple, `"abc def gh".split(" ")` s'évalue en `["abc", "def", "gh"]` et `"abc-def gh".split("-")` en `["abc", "def gh"]`.
  - Inversement, si `L` est une liste de chaînes et `c` une chaîne, `c.join(L)` s'évalue en le chaîne obtenue par réunification des chaînes de `L` via `c`. Par exemple, `"".join(["abc", "def", "gh"])` s'évalue en `"abcdefgh"` et `" truc ".join(["abc", "def", "gh"])` en `"abc truc def truc gh"`.
  - Du point de vue de l'accès, les chaînes se comportent comme des listes à ceci près qu'elles ne sont pas modifiables. Si `s` est une chaîne, `len(s)` est sa longueur et `s[i]` son i-ème caractère, pour $0 \leq i <$ `len(s)`.

Exercice 1. On se donne un fichier dont les premières lignes sont les suivantes :

```
Fichier mesures v1.0
x; y
0.0; 1.0
0.0317332591272; 0.9994965423831851
0.0634665182543; 0.9979866764718844
0.0951997773815; 0.9954719225730846
0.126933036509; 0.9919548128307953
```

Et il se poursuit sur un certain nombre de lignes de la forme `x; y`. On suppose que la deuxième colonne est fonction de la première, c'est-à-dire que y est de la forme $f(x)$ pour une certaine fonction f. Les abscisses sont croissantes. Le fichier s'appelle `mesures.txt` et se trouve dans le répertoire courant. Écrire un script affichant à l'écran la valeur approchée de l'intégrale de la fonction f sur l'intervalle compris entre la première abscisse et la dernière, obtenue avec la méthode des trapèzes.

Exercice 2. *Écriture dans un fichier*. (C'est essentiellement l'inverse de l'exercice précédent). Écrire une fonction `mesures(f, a, b, n, g)` prenant en entrée une fonction f, deux réels a et b, un entier $n > 0$, et un fichier `g` ouvert en écriture (`g` est un wrapper), et écrivant dans `g` des lignes de la forme $x; f(x)$. Il doit y avoir n lignes, avec les abscisses régulièrement espacées dans $[a, b]$.
Exemple avec `mesures(cos, 0, pi, 5, g)` :

```
0.0,1.0
0.785398163397,0.7071067811865476
1.57079632679,6.123233995736766e-17
2.35619449019,-0.7071067811865475
3.14159265359,-1.0
```

Exercice 3. *Extraction de lignes*. Écrire une fonction `extraire_nat(f,g)` prenant en entrée un fichier `f` ouvert en lecture, un fichier `g` ouvert en écriture. La fonction recopie dans `g` les lignes de `f` qui sont non vides (c'est-à-dire ne contenant pas un unique saut de ligne), et contenant uniquement un seul entier positif terminé par un saut de ligne. Voici un exemple :

Le fichier f

```
bla
42
28 55 3.14
-2
"coucou"
0
```

Le résultat dans g

```
42
0
```