TP : Boucles inconditionnelles : `for`

I) Chaînes de caractères

Exercice 1
Rappel : il faut toujours donner des noms pertinents aux fonctions, des noms qui explicitent ce que fait la fonction sans être non plus trop long.

1) Écrire une fonction qui recherche une chaîne de caractère (passée en argument) dans une autre chaîne de caractère.

2) Ouvrir le fichier Lecture-fichier.py, qui contient une fonction permettant d'ouvrir et de stocker dans une chaîne de caractères `Chaine` l'ensemble du fichier Exemple-texte.txt. Tester la fonction que vous venez de créer à la question précédente.

Vous pouvez aussi l'importer comme une bibliothèque, et utiliser la fonction comme une fonction de bibliothèque.

3) Recherche d'un motif.

a) À l'aide d'une fonction, comptez le nombre d'occurrence d'une chaîne de caractères donnée dans le texte. On pourra compter le nombre d'occurrence des mots 'passé', 'présent' et 'avenir'.

b) Faites afficher du contexte (en séparant la fonction affichage et la recherche du motif : votre fonction rendra une liste de chaînes de caractères pour chaque occurrence trouvée).

c) En déduire que la réponse à votre *vraie* question (trouver les mots passé / présent / avenir) n'était pas si simple. Proposer des pistes pour y remédier. Pensez à utiliser au maximum les listes.

4) (bonus) Comptez le nombre de mots dans le texte. Déterminer les mots présents, le nombre d'occurrence de chacun d'entre eux, et les classer par fréquence.

5) (bonus) Toujours à l'aide de fonctions, déterminer la plus longue phrase (en nombre de mots), et la longueur moyenne d'une phrase.

On utilisera la fonction `.split()` en coupant aux différentes ponctuations possibles. Il faudra aussi enlever les ponctuations parasites ('« ', ' »', '—', etc) qui, sinon, seront comptées comme des mots. Cf la documentation en ligne.

Style d'écriture d'un programme : on peut vouloir écrire les fonctions dans un fichier à part (par exemple dans le fichier `TP_string-lib.py`) et le corps du programme dans un second fichier (ex : `TP_string.py`) qui contient essentiellement des appels à des fonctions, qui ont été préalablement importée (`import`). Par contre, il n'est pas nécessaire de recommencer un nouveau fichier à chaque question, ni d'effacer la fonction précédente. La programmation à l'aide de fonction permet justement de laisser de coté certain bout de codes dont on se sert pas.

II) Listes

La boucle `for` permet de parcourir une liste (ou d'autre types de données, cf cours plus tard). Syntaxe :

```
for <variable> in <nom liste> :
      < bloc d'instructions du for >
```

Où la liste est une liste existante (on peut la créer avec `range`) et la variable est une variable locale à la boucle qui parcourt les éléments de la liste. Exemple :

```
for i in range(10) :
      print(2*i)
```

Ou bien :

```
L=[2,3.5,"vacances",True]
for x in L :
      print(type(x))
```

Exercice 2
Écrire une fonction qui calcule la norme d'un vecteur : elle prendra en entrée un n-uplet ou une liste, et rendra la racine carrée de la somme des carrés des termes de la liste : $\sqrt{x_1^2 + \cdots + x_n^2}$.

Exercice 3 (Recherche dans une liste)

1) Écrire une fonction qui recherche la présence d'un élément (passé en argument) dans une liste. La fonction rendra -1 si l'élément n'est pas présent, et l'indice de première occurrence sinon.

2) Écrire une fonction qui recherche le maximum d'une liste de nombre.

3) Écrire des fonctions qui calculent la moyenne et la variance d'une liste de nombres.

Exercice 4
Écrire une fonction qui prend en entrée une fonction f (de $\mathbb{R}$ dans $\mathbb{R}$), deux réels a et b, et un pas n, et qui calcule l'intégrale $\int_a^b f(t)\,\mathrm{d}t$ par la méthode des rectangles.
Même question pour la méthode des trapèzes.

III) Bonus

Exercice 5 (Tortue)
On pourra commencer par tester et comprendre le programme suivant :

```
import turtle as t
from math import pi

t.radians() # Parce que les gens biens parlent en radians
t.speed(0) # Pour que la tortue ne rame pas trop. cf l'aide.

def FonctionMystere(x): #
    for i in range(4):
        t.down()
        t.forward(x)
        t.left(pi/2)
        t.up()

for i in range(10): #
    FonctionMystere(50)
    t.left(pi/5)

t.goto(0,-100)
t.mainloop()
```

1) Rajouter les commentaires manquant devant la fonction, et lui donner un nom explicite !!

2) **a)** Tracer le dessin que vous voulez. Puis reproduisez-le en plusieurs exemplaires.

b) Adapter vos différentes fonctions pour pouvoir reproduire votre dessin à plusieurs échelles différentes. (on pourra créer un effet de perspective).

Morale : passer des valeurs numérique directement dans le corps d'un programme est Mal. Les variables, c'est Bien : il faut toujours, autant que possible, stocker les valeurs dans des variables puis appeler ces variables.

3) **a)** Programmer une fonction ayant deux arguments qui dessine un polygone régulier.

b) Programmer une fonction qui répète n fois un motif (qui sera passé en argument).