Pcsi Interrogation écrite

EXERCICE 1: Une simulation

Q1: Ecrire une fonction `moyenne(L)` qui renvoie la moyenne des éléments d'une liste `L` de nombres.

Q2: Si `L` est une liste de nombres dont la moyenne des éléments est `M`. On dit que `L` est proche de sa moyenne à `p` près si et seulement l'inégalité $|L[i] - M| \leq p$ est vraie pour tout élément $L[i]$ de la liste `L`.
Ecrire une fonction `proche_moyenne(L,p)` qui renvoie `True` si `L` est proche de sa moyenne à `p` près et `False` sinon (on pourra utiliser la fonction **abs** de python).

La fonction random de la bibliohèque random renvoie un nombre réel appartenant à $[0, 1[$ tiré au sort.

```
import random
x=random.random() # affecte a x un nombre tire au sort entre 0 et 1 (loi uniforme)
```

Si x est obtenu en appliquant la fonction random et a et b sont deux réels, alors $a + (b - a)\,x$ est un nombre tiré au sort entre a et b.

Soit L une liste de nombres de moyenne M. La liste suivante de L est une liste LL de même longueur que L formée de nombres tirés au sort de la manière suivante:

Pour tout i, $LL[i]$ est tiré au sort entre $L[i]$ et M.

Q3: Ecrire une fonction `suivante(L)` qui renvoie une liste suivante de la liste `L` obtenue en utilsant la fonction random.

Un liste `L` étant donnée, on considère la suite de listes définies par $L_0 = L$ et L_{i+1} est la liste obtenue de L_i par la fonction suivante. On appelle alors temps de convergence à `p` près la plus petite valeur de i pour laquelle L_i est proche de sa moyenne à `p` près.

Q4: Ecrire une fonction `temps_convergence(L,p)` qui renvoie le temps de convergence à `p` près lorsque L_0 vaut `L`.

On cherche à estimer le temps de convergence moyen lorsque `L` est une liste de `n` réels tirés au hasard entre deux réels `a` et `b` . Pour cela, on se propose de calculer la moyenne de plusieurs temps de convergence obtenus avec la fonction précédente.

Q5: Ecrire une fonction `temps_convergence_moyen(a,b,n,N,p)` qui renvoie la moyenne des temps de convergences à `p` près de `N` listes de longueur `n` formés d'éléments tirées au hasard entre `a` et `b`.

EXERCICE 2: chaînes de caractères

On se donne une liste de caractères appelée `séparateurs` et une chaîne de caractères `texte`. Un mot de `texte` est une suite de caractères consécutifs de `texte` encadrés par deux caractères de `séparateurs` (ou seulement délimitée à droite par un caractère de `séparateurs` en début de `texte`. Par exemple, si `separateurs=[' ',',','.','?']` (autrement dit la liste des caractères espace, virgule, point et point d'interrogation) et
`texte='Bonjour, où est le cours?  Je suis perdu.'` La liste des mots de texte est
`['Bonjour','où','est','le','cours','Je','suis','perdu']`.

Q6: Ecrire une fonction `liste_mots(texte,separateurs)` qui renvoie la liste des mots de `texte`.

Q7: Dans la fonction suivante, on suppose que `positions` est une liste d'entiers de longueur égale au nombre k de mots de `texte` et telle que `position[i]` appartient à $[[0, k-1]]$ pour tout i.
Ecrire une fonction `melange(texte,separateurs,positions)` qui renvoie la chaine de caractères dont le $\texttt{i}^{ème}$ mot est le $\texttt{j}^{ème}$ mot de texte avec `j=positions[i]`. Les mots étant simplement séparés par des espaces.
Avec les valeurs précédentes,
`melange(texte,separateurs,[0,2,4,6,1,3,5,7])`→`'Bonjour est cours suis où le Je perdu '`
(on finit le dernier mot avec le séparateur espace)

Q8: Ecrire une fonction `remet_dans_ordre(texte_melange,positions)` qui renvoie le texte dans l'ordre (avec uniquement des espaces comme séparateurs)
Si.`texte_melange='Bonjour est cours suis où le Je perdu'`
`remet_dans_ordre(texte_melange,[0,2,4,6,1,3,5,7])`→`'Bonjour où est le cours Je suis perdu.'`

CORRECTION:

```
def moyenne(L):
    S=0
    for i in range(len(L)):
        S=S+L[i]
    return S/len(L)

def proche_moyenne(L,p):
    M=moyenne(L)
    res=True
    for i in range(len(L)):
        if abs(L[i]-M)>p:
            res=False # il suffit d'un cas pour que le resultat soit faux
    return res

import random
def suivante(L):
    a=moyenne(L)
    LL=[]
    for i in range(len(L)):
        x=random.random()
        b=L[i]
        y=a+(b-a)*x
        LL.append(y)
    return LL

def temps_convergence(L,p):
    i=0
    while(proche_moyenne(L,p))==False:
        L=suivante(L)
        i=i+1
    return i

def temps_convergence_moyen(a,b,n,N,p):
    T=0
    for i in range(N):
        L=liste_hasard(n,a,b)
        T=T+temps_convergence(L,p)
    return T/N

def liste_hasard(n,a,b):
    L=[]
    for i in range(n):
        x=random.random()
        y=a+(b-a)*x
        L.append(y)
    return L
```

Dans la fonction suivante, il faut tenir compte du fait qu'on peut avoir deux séparateurs consécutifs.
Si vous recopiez ce programme pour le faire tourner su python, réécrivez les mots vides.

```
def liste_mots(texte,separateurs):
    liste=[]
    mot=texte[0] # le texte commece par un caractere
    for i in range(1,len(texte)):
        if est_element(separateurs,texte[i]):
            if mot!='':# premier separateur rencontre apres le mot
                liste.append(mot)
                mot='' # si on rencontre un deuxieme separateur, le mot est vide
        else:
            mot=mot+texte[i]
    return liste

def est_element(L,x):
    for i in range(len(L)):
        if L[i]==x:
            return True
    return False

separateurs=[' ',',','.','?']
texte='Bonjour, o\U{f9} est le cours? Je suis perdu.'

def melange(texte,separateurs,positions):
    L=liste_mots(texte,separateurs)
    s=''
    for i in range(len(L)):
        s=s+L[positions[i]]
        s=s+' '
    return s
```

Pour cette dernière question, on remarque la chose suivante: Par le mélange, le $i^{ème}$ mot de texte est devenu le $j^{ème}$ avec j=position[i]. Pour la remise en ordre, Il faut donc mettre en le $j^{ème}$.mot en $i^{ème}$. Il s'agit de la même opération que dans mélange avec une nouvelle liste de positions, positions_retour verifiant positions_retour[j]=i. On a positions_retour[j]=i⇔positions[i]=j (mathématiquement, il s'agit de la réciproque).

```
def remet_dans_ordre(texte_melange,separateurs,positions):
    positions_retour=retour(positions)
    return melange(texte_melange,separateurs,positions_retour)

def retour(positions):
    L=[]
    n=len(positions)
    for i in range(n):
        for j in range(n):
            if positions[j]==i: # un seul j le verifie.
                L.append(j)
    return L
```