Corrigé de la feuille d'exercices 2

Exercice 1

```
## Exercice 1 : Moyenne et Variance

# 1) Moyenne version 1
def moyenne(L):
    card = 0
    somme = 0
    for x in L:
        somme += x
        card += 1
    return somme/card

# Moyenne version 2
def moyenne1(L):
    return sum(L)/len(L)

# 2) Variance version 1
def variance(L):
    L2 = []
    for x in L:
        L2.append(x**2)
    return moyenne(L2)-moyenne(L)**2

# Variance version 2
def variance1(L):
    L2 = [ x**2 for x in L]
    return moyenne(L2)-moyenne(L)**2
# Variance version 3
def variance2(L):
    m = moyenne(L)
    LL = [ (x-m)**2 for x in L ]
    return moyenne(LL)
```

Test :

```
# 3)
from random import uniform
a, b = 0, 10
X = [ uniform(a,b) for k in range(10000) ]
print("Moyenne :",moyenne(X))
print("Variance :",variance(X))
```

Exercice 2

```
## Exercice 2 : Recherche d'espace libre contigu
# 1)
def nombreZeros(t,i):
    n = len(t)
    r = 0
    while i < n and t[i] == 0:
        i += 1
        r += 1
    return r

# 2)
def nombreZerosMax(t):
    n = len(t)
    L = [nombreZeros(t,i) for i in range(n) ]
    return max(L)
# 3)
```

```
def nombreZerosMax1(t):
    ListeNombreZeros = []
    i = 0
    while i < len(L):
        d = nombreZeros(t,i)
        ListeNombreZeros.append(d)
        i += d+1
    return max(ListeNombreZeros)
```

Exercice 3

```
## Exercice 3. Cryptage de Cesar.

# 1)
alphabet = 'abcdefghijklmnopqrstuvwxyz'

# 2)
def decalage(n):
    resultat = ''
    for k in range(26):
        resultat += alphabet[(k+n)%26]
    return resultat

# 3)
def indices(x,phrase):
    n = len(phrase)
    L = []
    for i in range(n):
        if x == phrase[i]:
            L.append(i)
    return L

# 4)
def codage(n,phrase):
    alphabetCode = decalage(n)
    listePhrase = []
    for x in phrase:
        listePhrase.append(x)
    for i in range(26):
        lettre = alphabet[i]
        lettreCode = alphabetCode[i]
        ind = indices(lettre,phrase)
        for i in ind:
            listePhrase[i] = lettreCode
    phraseCode = ''
    for l in listePhrase :
        phraseCode += l
    return phraseCode

# 5)
def decodage(n,phrase):
   return codage(-n,phrase)
```

Pour la fonction `codage` *: créer une liste* `listePhrase` *des caractères de la chaine* `phrase` *; parcourir les 26 lettres de l'alphabet par indice, en appelant les fonctions* `indices(x,phrase)` *et* `decalage(n)`*, modifier chaque lettre* `x` *dans* `listePhrase` *par le caractère codé correspondant. Finalement reconstituer à partir de* `listePhrase` *la phrase codée.*

Exercice 4

```
## Exercice 4. Parcours de graphe.

# 1)
M =[[0, 9, 3, -1, 7],
    [9, 0, 1, 8, -1],
    [3, 1, 0, 4, 2],
    [-1, 8, 4, 0, -1],
    [7, -1, 2, -1, 0]]

# 2)
Voisins4 = []
for i in range(5):
    if M[4][i] != -1 and M[4][i] != 0:
        Voisins4.append(i)

# 3)
def voisins(i):
    vois = []
    for j in range(5):
        # M etant une variable globale on peut y acceder
        # en lecture au sein d'une fonction
        if M[i][j] != -1 and M[i][j] != 0:
            vois.append(j)
    return vois

# 4)
def degre(i):
    return len(voisins(i))
```

```
# 5)
def longueur(L):
    longueur = 0
    origine = L[0]
    for i in range(1,len(L)):
        extremite = L[i]
        if extremite not in range(5)
                        or M[origine][extremite] == -1 :
            return -1
        longueur += M[origine][extremite]
        origine = extremite
    return longueur
```