Informatique en CPGE (2018-2019)
Corrigé TP 6 : les types composés

Exercice 1

Questions 1, 2 et 3.

```
from random import randint

def chaine(n):
    dna=''
    for i in range(1000):
        alea=randint(1,4)
        if alea==1:
            dna+='A'
        elif alea==2:
            dna+='C'
        elif alea==3:
            dna+='G'
        else:
            dna+='T'
    return dna

# ou bien
def chaine(n):
    choix='ACGT'
    for i in range(1000):
        alea=randint(0,3)
        dna+=choix[alea]
    return dna

dna=chaine(1000)

def compteurACGT(chaine):
    cptA,cptC,cptG,cptT=0,0,0,0
    for car in chaine:
        if car=='A':
            cptA+=1
        elif car=='C':
            cptC+=1
        elif car=='G':
            cptG+=1
        else:
            cptT+=1
    return (cptA,cptC,cptG,cptT)

print(compteurACGT('AAACGGTCGA')  # test sur une chaîne courte
print(compteurACGT(dna))
```

Question 4

```
from random import randint

def chaine(n):
    dna=[]
    for i in range(1000):
```

```
        alea=randint(1,4)
        if alea==1:
            dna.append('A')
        elif alea==2:
            dna.append(C')
        elif alea==3:
            dna.append('G')
        else:
            dna.append('T')
    return dna

# ou bien
def chaine(n):
    choix='ACGT'
    for i in range(1000):
        alea=randint(0,3)
        dna.append(choix[alea])
    return dna

# ou bien
def chaine(n):
    choix='ACGT'
    return [choix[randint(0,3)] for i in range(1000)]
```

Le reste est identique.

Exercice 2

```
def chiffre(chaine,key):
    c=''
    for i in range(len(chaine)):
        if 65<=ord(chaine[i])<=90: # pour les majuscules
            code=(ord(chaine[i])-65+key)%26+65
            c+=chr(code)
        elif 97<=ord(chaine[i])<=122: # pour les minuscules
            code=(ord(chaine[i])-97+key)%26+97
            c+=chr(code)
        elif 48<=ord(chaine[i])<=57: # pour les chiffres de 0 à 9
            code=(ord(chaine[i])-48+key)%26+48
            c+=chr(code)
        else:
            c+=chaine[i]
    return c
print(chiffre('bonjour',3))
print(chiffre('BONJOUR',7))

def dechiffre(chaine,key):
    c=''
    for i in range(len(chaine)):
        if 65<=ord(chaine[i])<=90:
            code=(ord(chaine[i])-65-key)%26+65
            c+=chr(code)
        elif 97<=ord(chaine[i])<=122:
            code=(ord(chaine[i])-97-key)%26+97
            c+=chr(code)
        elif 48<=ord(chaine[i])<=57:
            code=(ord(chaine[i])-48-key)%26+48
            c+=chr(code)
```

```
        else:
            c+=chaine[i]
    return c
print(dechiffre(chiffre('Bonjour',3),3))

# ou bien ?
def dechiffre(chaine,key):
    return chiffre(chaine,-key)
```

Exercice 3

```
def moyenne(liste):
    n=len(liste)
    s=0 # on initialise une somme à zéro
    for i in range(n):
        s+=liste[i] # les termes de la liste sont additionnés
    return s/n

# ou bien
def moyenne(liste):
    s=0 # on initialise une somme à zéro
    for x in liste:
        s+=x # les termes de la liste sont additionnés
    return s/len(liste)

def minimum(liste):
    mini=liste[0] # on prend comme minimum au départ le premier terme
    for i in range(len(liste)):
        if liste[i]<mini:
            mini=liste[i] # on modifie le minimum si le terme étudié
            # est plus petit que celui enregistré
    return mini

def maximum(liste): # sans les indices qui ne sont pas utiles
    maxi=liste[0] # on prend comme maximum au départ le premier terme
    for x in liste:
        if x>maxi:
            maxi=x # on modifie le maximum si le terme étudié
            # est plus grand que celui enregistré
    return maxi

def mediane(liste):
    n=len(liste)
    liste.sort()
    if n%2==1 :
        med=liste[(n-1)//2] # on calcule la médiane selon que l'effectif
        # est pair ou non
    else:
        med=(liste[n//2-1]+liste[n//2])/2
    return med

def statistiques(liste):
    moy=moyenne(liste)
    mini=minimum(liste)
    maxi=maximum(liste)
    med=mediane(liste)
    return moy,mini,maxi,med
```

Exercice 4

```
def produit(m1,m2):
    p=[[],[]]
    p[0].append(m1[0][0]*m2[0][0]+m1[0][1]*m2[1][0])
    p[0].append(m1[0][0]*m2[0][1]+m1[0][1]*m2[1][1])
    p[1].append(m1[1][0]*m2[0][0]+m1[1][1]*m2[1][0])
    p[1].append(m1[1][0]*m2[0][1]+m1[1][1]*m2[1][1])
    return p

# ou bien
def produit(m1,m2):
    p=[[0,0],[0,0]]
    for i in range(2):
        for j in range(2):
            for k in range(2):
                p[i][j]+=m1[i][k]*m2[k][j]
    return p

mat1=[[0,1],[2,3]]
mat2=[[4,5],[2,3]]
print("matrice produit AB : ",produit(mat1,mat2))
print("matrice produit BA : ",produit(mat2,mat1))
```

```
def permuteLignes(m):
    return [m[1],m[0]]

def permuteColonnes(m):
    L1=[m[0][1],m[0][0]]
    L2=[m[1][1],m[1][0]
    return [L1,L2]
```