Le chiffre de Vigenère et les carrés magiques (Correction)

1. Le chiffre de Vigenère

Q.1. Initialisation avec des variables globales

```
global alphabet,N,cle,L
alphabet= "abcdefghijklmnopqrstuvwxyz"
N=len(alphabet)
cle="roue"
L=len(cle)
```

Q.2. La méthode de chiffrement

```
def code_vigenere(ch):
    code=""
    for i in range(len(ch)):
        d=i
        while d>L-1:
            d=d-L
        index=alphabet.index(ch[i])+alphabet.index(cle[d])
        if index>N-1:
            index=index-N
        j=alphabet[index]
        code=code+j
    return code
```

Q.3. La méthode de chiffrement

```
def decode_vigenere(ch):
    decode=""
    for i in range(len(ch)):
        d=i
        while d>L-1:
            d=d-L
        index=alphabet.index(ch[i])-alphabet.index(cle[d])
        if index<0:
            index=index+N
        j=alphabet[index]
        decode=decode+j
    return(decode)
```

Q.4. L'intérêt de cette méthode

Le chiffre de Vigenère est une amélioration de la méthode de César, son principal intérêt réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message que l'on retrouve dans le carré de Vigenère (d'où l'appellation polyalphabétique). Il est ainsi bien plus difficile à casser que celui de César, on passe d'une clé sous la forme d'un nombre entier de $d \in [1,25]$ à une clé sous la forme d'une chaine de caractère de longueur inconnue. Toutefois 300 ans après sa création plusieurs techniques permettant de caser cette méthode de chiffrement ont été développées.

2. Les carrés magiques

Q.1. Carré magique complété sans l'aide du programme (taille 5) :

15	22	9	16	3
2	14	21	8	20
19	1	13	25	7
6	18	5	12	24
23	10	17	4	11

On vérifie les trois propriétés d'un carré magique avec La somme de chaque colonne, la somme de chaque et la somme de chaque diagonale est égale à la densité $d = 65$.

Q.2. Création du carré vide :

```
from copy import *
def Carre_vide(n):
    if n%2==0:
        print("Erreur n doit être impair")
    else:
        carre=[]
        for i in range(n):
            carre.append([0]*n)
    return carre
```

Q.3. Méthode de remplissage :

```
def Remplir_carre(carre):
    n=len(carre)
    carre_magique=deepcopy(carre)
    x,y=(n-1)//2-1,(n-1)//2
    for i in range(1,n**2+1):
        carre_magique[y][x]=i
        print(carre_magique)
        if i%(n)==0:
            x=(x-2)%n
        else:
            x,y=(x-1)%n,(y-1)%n
    return carre_magique
```

Q.4. Méthode de vérification :

On vérifie les trois propriétés d'un carré magique avec les sommes suivantes égales à la densité :

- La somme de chaque colonne
- La somme de chaque ligne
- La somme de chaque diagonale

```
def Verif_carre(carre_magique):
    n=len(carre)
    dens=int(n*(n**2 + 1)/2)
    for i in range(n):
        d1=0
        d2=0
        for j in range(n):
            d1+=carre_magique[j][i]
            d2+=carre_magique[i][j]
        if d1!=dens or d2!=dens:
            return False
    d1=0
    d2=0
    for i in range(n):
        d1+=carre_magique[i][i]
        d2+=carre_magique[n-1-i][i]
    if d1!=dens or d2!=dens:
        return False
    return True
```