```
from scipy.integrate import odeint
import matplotlib.pyplot as pl
import numpy as np
from math import *
#**************Question 1*****************************
# on a yn+1=yn+h*phi(t,y,h)
# c'est à dir : yn+1=yn+h*alpha*f(t,y)+beta*f(t+h,y+h*f(t,y))
# pour euler : yn+1=yn+h*f(t,y)
# ce qui implique qu'on prendra alpha=1 et beta=0
#**********************Question2************************
def Heun(f,t0,y0,T,N):
    h=T/N
    tt=[t0+i*h for i in range(N+1)]
    Y=[]
    y=y0
    for t in tt:
        k1=f(t,y)
        k2=f(t+h,h*k1)
        y+=h*(k1+k2)/2
        Y.append(y)
    return tt,Y
#***************Question 3*****************************
# U''(t)+U'(t)+u(t)=-4*pi**2*cos(2*pi*t)
# on pose U''(t)=-U'(t)-U(t)-4*pi**2*cos(2*pi*t)
# on pose Y(t)=(U(t),U'(t)) donc Y'(t)=(U'(t),U''(t))
# on aura donc : Y'(t)=A(t)*Y(t)+B(t)
# Avec A(t)=[[0,1][-1,-1]] et B(t)=[0,-4*pi**2*cos(2*pi*t)]
```

```
def Euler(f,t0,T,y0,N):
    h=T/N
    tt=[t0+i*h for i in range(N+1)]
    y=y0
    Y=[]
    for x in tt:
        y=y+h*f(y,x)
        Y.append(y)
    return tt,Y
#**************************Question 4************************************
#si on prend N comme facteur de complexité alors la complexité de cette fonction est N multiplié par la complexité de f(x,y) : d'ou c(N)=O(N)*c(f(x,y))


#*********************************Question 5**************************
#En prenant y(t)=U(t) et z(t)=U'(t) on aura l'équation 1 comme :
#[y'(t),z'(t)]=[[0,1],[-1,-1]]*[y(t),z(t)]+[0,-4*pi**2*cos(2*pi*t)]
#d'où le système :      y'(t)=z(t)
#                et
#                    z'(t)=-y(t)-z(t)-4*pi**2*cos(2*pi*t)
# d''ou on a deux fonctions auxquels on appliquera euler ou heun


#**************************Question 6:**************************
#en posant Y(t)=[U(t),U'(t)]
#on a Y'(t)=[U'(t),U''(t)]
# on peut donc avoir l'equation comme suit:
# Y'(t)=[[0,1],[-1,-1]]*Y(t)+[0,-4*pi**2*cos(2*pi*t)]
# donc F(t,X)=A(t)X+ B(t) avec A(t)=[[0,1],[-1,-1]] et  B(t)=[0,-4*pi**2*cos(2*pi*t)]
# **************************question 7********************
```

```
def F(Y,t):  # fontion utilisée dans Euler vectorielle
  return np.array([Y[1],-Y[0]-Y[1]-4*pi**2*cos(2*pi*t)])
#*********************************************************
def F2(Y,Z,t): #première fonction du système de la question 5:fonction  y'(t)=z(t) =F2(y(t),z(t),t)
   return Z
#*********************************************************
def F3(Y,Z,t): #deuxième fonction du système de la question 5:fonction  z'(t)=-y(t)-z(t)-
4*pi**2*cos(2*pi*t) = F3(y(t),z(t),t)
   return -Y-Z-4*pi**2*cos(2*pi*t)
#*********************************************************
def Heun1(G,H,t0,y0,y10,T,N):
   """la fonction de heun utilisant les fonctions y(t) et z(t) du systeme de la question 5"""
   h=T/N
   tt=[t0+i*h for i in range(N+1)]
   Y=y0
   Z=y10
   LY=[[Y,Z]]
   for i in range (1,len(tt)):
      t=tt[i]
      k1y=G(Y,Z,t)
      k1z=H(Y,Z,t)
      k2y=G(Y+h*k1y,Z+h*k1z,t+h)
      k2z=H(Y+h*k1y,Z+h*k1z,t+h)
      Y=Y+h*(k1y+k2y)/2
      Z=Z+h*(k1z+k2z)/2
      LY.append([Y,Z])

   return LY
```

```
#*****************************************************

################  Décommenter pour tester le code du calcul scientifique   ############""

# N=1000

# T=3

# t0=0

# U0=0      #U(0)

# U10=0.1    #U'(0)

# y0=[U0,U10]

# h=T/N

# tt=[i*h for i in range(N+1)] #liste des abscices temps

# YY=odeint(F,y0, tt)           # resultat de l'application de Odeint

# zz=Heun1(F2,F3,0,0,0,T,N)      # resultat de l'application de Heun

# rr=np.array(zz)       #transformation de la liste de listes résultat zz en numpy array

# HH1=Euler(F,t0,T,y0,N)         # resultat de l'application d'Euler

# HH=np.array(HH1[1])   # récupération des Y et transformation de la liste de listes en numpy array

# pl.title("Circuit RLC, tension Bobine")     # titre de la figure

# pl.xlabel("temps(s)")                  # texte de l'axe des x

# pl.ylabel("tension(mV)")                # texte de l'axe des y

# pl.plot(tt,YY[:,0],label='odeint')         #tracé de odeint et sa legende

# pl.plot(tt,rr[:,0],label='Heun')          #tracé de Heun et sa legende

# pl.plot(tt,HH[:,0],label='Euler')         #tracé de Euler et sa legende

# pl.legend()                          # affichage de la légende

# pl.show()                           # affichage global des courbres


################  Fin du problème I ##########################


########################## début du problème II  ###############
```

```
#_____Question 8_____#
def changeAccent(c):
    """ cette fonction transforme un caractère accentué en argument en son equivalent normal"""
    alphaAcc="âàéèêëïîôùûüÿç"
    c1=c
    if c in alphaAcc:
        if c in "âà":
            c1="a"
        elif c in "éèêë":
            c1="e"
        elif c in  "ïî":
            c1="i"
        elif c=="ô":
            c1="o"
        elif c in "ùûü":
            c1="u"
        elif c=="ÿ":
            c="y"
        elif c=="ç":
            c1="c"
    return c1

#*****************************************************************#
def enMajuscule(CH):
    """cette fonction transorme une phrase en minuscule accentuée ou non en son equivalente
    majuscule sans tenir compte des symboles, chiffre et ponctuations"""
    alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
    alphaAcc="âàéèêëïîôùûüÿç"
    alphaMin="abcdefghijklmnopqrstuvwxyz"
    CH1=""
    for x in CH:
        cMin=changeAccent(x)
        if cMin in alphaMin:
            pos=alphaMin.index(cMin)
            cMaj=alphaMaj[pos]
            CH1+=cMaj
        else:
            CH1+=cMin
    return CH1
#*********************************************************************#
#_____________________________________Question 9_____________________#
def majusculesSeules(CH):
    """ cette fonction utilise la précedente et supprime tous les symboles, espaces et ponctuation"""
    CH1=""
    chMaj=enMajuscule(CH)
    alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    for x in chMaj:
        if x in alphaMaj:
            CH1+=x
    return CH1
#*********************************************************************#
#_____________________________________Question 10____________________#
def vigenereEncode(CH,CL):
    """ fonction de codage vigenere """
```

```
    alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    chCle=CL*len(CH) # même si la chaine chCle est de taille suprieure à celle de CH
                    # c'est le traitement de CH qui gere le résultat
                    # on pouvait faire chCle=CL*(len(CH)//len(CL))+CL[:len(CH)%len(CL))]
    chInitiale=majusculesSeules(CH)
    chResultat=""
    for i in range(len(chInitiale)):
        posIni =alphaMaj.index(chInitiale[i])
        posCle =alphaMaj.index(chCle[i])
        posRes=(posIni+posCle)%26
        chResultat+=alphaMaj[posRes]
    return chResultat
#*****************************************************************#
#_____________________________________Question 11_____________________#
# voir ce lien pour plus d'information : https://wiki.python.org/moin/TimeComplexity
# sachant don que la complexité de l'indiçage est O(1)
# la complexité de index au pire est (n)   (voir le lien précédent)
# la complexité de la fonction précédente :
# voyons la complexité des fonctions : changeAccent-enMajuscule-majusculesSeules
# 1- la complexité de changeAccent dans le pire des cas est la complexit de recherche dans la liste
# des accents :"âàéèêëïîôùûüÿç"
# donc c(changeAccent) est O(14)
# 2- la complexité de enMajuscule  :si m est la longueur de la chaine CH
# alors on a une recherche de caractère minuscule dans la chaine  alphaMin et recupération par index
# de la position ce qui fait au pire des cas :26*26 qui tout de meme constante
# donc  la complexité de  enMajuscule est : O(m*26*26 )+O(m*14)
# 3- la complexité de majusculesSeules est : une boucle de taille m avec recherhce dans alphaMaj
```

```
# donc la complexité est :  O(m*26)+O(m*26*26 )+O(m*14)
# la fonction précedente contient une boucle for qui depend de la taille de chInitiale m,
# la meme taille que CH + complexité de la creation de chCle qui est O(m*p) avec p =len(CL)
# la complexité de la fonction majusculesSeules est :O(m*26)+O(m*26*26 )+O(m*14)
# la boucle for contient deux recherches index ca fait donc :(m+p*m)*m
# d'ou la complexité globale est ((m+p*m)*m)+O(m*26)+O(m*26*26 )+O(m*14)+O(m*p)
# donc :c(m)=O(m²p)+O(m)+O(m*p)=0(m²*p) si p est fixe : c'est O(m²)

#*********************************************************************#
#_____________________________________Question 12_____________________#
def vigenereEncodeRec(CH,CL,indx,indxCl):
    alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    if indx==len(CH):
        return ""
    else:
        c=CL[indxCl]
        indxCl+=1
        if indxCl>=len(CL):
            indxCl=0
        nouvCar=alphaMaj[(alphaMaj.index(CH[indx])+alphaMaj.index(c))%26]
        indx+=1
        return nouvCar+vigenereEncodeRec(CH,CL,indx,indxCl)
#***********************************
#_____________________________________Question 13_____________________#
# la complexité de la fonction précédente :
# la fonction précedente contient un appel recursif
#si len(CH)=m alors la complexité de la fonction est :
```

```
# complexité de vigenereEncodeRec est :c(i)=26+c(i+1)+1 et c(m)=0
# c(m)=27*m
# c(m)=O(m)
#
#
#*********************************************************************#
#_____________________________________Question 14_____________________#
def vigenereDecode(CH,CL):
    """ fonction de décodage vigenere """
    alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    chCle=CL*len(CH) # même si la chaine chCle est de taille suprieure à celle de CH
                     # c'est le traitement de CH qui gere le résultat
                     # on pouvait faire chCle=CL*(len(CH)//len(CL))+CL[:len(CH)%len(CL))]
    #chInitiale=majusculesSeules(CH)
    chResultat=""
    for i in range(len(CH)):
        posIni =alphaMaj.index(CH[i])
        posCle=alphaMaj.index(chCle[i])
        posRes=(posIni-posCle)%26
        chResultat+=alphaMaj[posRes]
    return chResultat
#*********************************************************************#
#_____________________________________Question 14_____________________#
# Le choix de la clé est important : il faut qu'il soit un peu long et constitué de caractères non
# consécutifs
texte="Hey Hey, cet été là, le gang des niçois a été arrêté!"
cch=vigenereEncode(texte,"ABC")
```

```
print(cch)
print(vigenereEncodeRec("ABCDEF","ABC",0,0))
print(vigenereDecode(cch,"ABC"))
#################  Fin du problème II ##########################


######################### début de la seconde partie  ###############
#
#*********************************************************************#
#_____________________________________Question 16_____________________#
def sousChaine(CH,d,p):
   return CH[d::p]
#*********************************************************************#
#_____________________________________Question 17_____________________#
def listeDesSousChaines(CH,d):
   return [sousChaine(CH,i,d) for i in range(d)]
#*********************************************************************#
#_____________________________________Question 18_____________________#
def frequencecaracteres(CH):
   alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
   frequence=[0]*26
   for i in range(len(CH)):
      frequence[alphaMaj.index(CH[i])]+=1
   return frequence
#*********************************************************************#
#_____________________________________Question 19_____________________#
def code(CH,p):
   alphaMaj="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
    L=listeDesSousChaines(CH,p)
    chCode=""
    for ech in L:
        Lf=frequencecaracteres(ech)
        posMax=Lf.index(max(Lf))
        posReel=(posMax-alphaMaj.index('E'))%26
        car=alphaMaj[posReel]
        chCode+=car
    return chCode



#*****************************************************************#
print(sousChaine(cch,1,3))
print(listeDesSousChaines(cch,3))
print(frequencecaracteres(cch))
print(code(cch,3))
```