```python
# -*- coding: utf-8 -*-
"""
Récapitulatif résolution des équations différentielles d'ordre 2 avec Python
"""
#On se ramène à une équa diff d'ordre 1 dY/dt=F(Y,t) où on définit le vecteur
#Y de coordonnées (y,dy/dt)
#F étant la fonction qui relie le vecteur dY/dt de coordonnées(dy/dt,d2y/dt2)
#au vecteur Y(y, dy/dt) avec la condition initiale Yo(y(0),dy/dt(0))
#on utilise alors la méthode d'Euler ou odeint(de préférence) vues précédemme
#pour l'ordre 1 ou directement la solution analytique si connue
import numpy as np #pour la manipulation de tableaux, construction histogramm
# avec hist, génération de nombres aléatoire avec random, régression linéaire
#avec polyfit,fonctions mathématiques et nombre pi
import matplotlib.pyplot as plt #pour les représentations graphiques
from scipy.integrate import odeint #pour la résolution d'équations différenti
#les

def eulerV(f,xo,t):#fonction euler vectorielle, f fonction vectorielle,xo
#vecteur des CI et t tableau des dates ti
    N=len(t)
    h=t[1]-t[0]
    x=np.zeros((2,N))#tableau d'initialisation 2 lignes et N colonnes de zéro
    x[:,0]= xo     #on applique le vecteur CI aux valeurs de la colonne 0
    for i in range(N-1):
        x[:,i+1]=x[:,i]+h*f(x[:,i],t[i])# on calcule successivement le vecteu
        #x aux différentes dates ti
    return x
#exemple mouvement oscillatoire d'un pendule simple ou pesant
#d2theta/dt2=-sintheta avec omega0 au carré=1     equa diff non linéaire
#pas de sln analytique connue
#résolution euler
def fV(X,t):#theta=X[0] et dtheta/dt=X[1]
    d1=X[1]
    d2=-np.sin(X[0])
    return np.array([d1,d2]) #renvoie dX/dt (dthteta/dt, d2theta/dt2=-sinthet
to=0
tf=2*np.pi*2 #omegaoau carré =1 d'où To=2pi, on prend tf=2 périodes
N=10000
T=np.linspace(to,tf,N+1)
X0=np.array([np.pi/18,0.])
sole=eulerV(fV,X0,T)
th=sole[0,:]*180/np.pi
plt.figure()
plt.plot(T,th)
plt.xlim(0,12)
plt.ylim(-10,10)
plt.show()
#résolution odeint
sol=odeint(fV,X0,T)
theta=sol[:,0]*180/np.pi
plt.figure()
plt.plot(T,theta)
plt.xlim(0,12)
plt.ylim(-10,10)
plt.show()

#résolution euler autre version
def euler(f,yo,t):
     N=len(t)
     y=yo
```

```
    L=[y]
    for i in range (N-1):
        y=y+f(y,t[i])*(t[i+1]-t[i])
        L.append(y)
    return np.array(L)
sole=euler(fV,X0,T)
th=sole[:,0]*180/np.pi
#graphe
plt.figure()
plt.plot(T,th)
plt.xlim(0,12)
plt.ylim(-10,10)
plt.show()
```