```
#Exemples commandes

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

#Stabilité différence finie, Dx est le coeff de diffusion
if dt < dx**2/(2*Dx):
    print("OK, schéma numérique stable en x.")

#Exemple Euler
def forward_euler(f, u0, T, N):
    #Solve u'=f(t, u), u(0)=u0, with n steps until t=T.
    t = np.zeros(N + 1)
    u = np.zeros(N + 1) # u[n] is the solution at time t[n]
    u[0] = u0
    dt = T / N
    for n in range(N):
        t[n + 1] = t[n] + dt
        u[n + 1] = u[n] + dt * f(t[n], u[n])

    return t, u

#Exemple : implémentation de l'équation différentielle f'(u) = u, avec u(0) = 1, sur un temps t = 4.
#On s'attend bien sûr à une croissance exponentielle

def f(t, u):
    return u

u0 = 1
T = 4
```

```
N = 30

t, u = forward_euler(f, u0, T, N)

fig, ax = plt.subplots()
ax.plot(t, u)
plt.show()

#Exemple de résolution d'un système EDO avec solve_ivp
def dA(Ca, t):
    return -k1*Ca*Cb + km1*Cc
def dB(Ca, Cb, Cc, t):
    return -k1*Ca*Cb + km1*Cc
def dD(Cc, Cd, t):
    return -k2*Cc*Cd
def dE(Ca, Cb, Cd, t):
    return k1*Ca*Cb/(1+km1/(k2*Cd))

def Cc(Ca, Cb, Cd):
    return k1*Ca*Cb/(km1 + k2*Cd)

k1 = 2
km1 = 1
k2 = 0.5
Ca0 = 5
Cb0 = 2
Cd0 = 1
Ce0 = 0
n = 10

from scipy.integrate import solve_ivp
```

```
def rhs(t, C):
    return [-k1*C[0]*C[1] + km1*Cc(C[0],C[1],C[2]), -k1*C[0]*C[1] + km1*Cc(C[0],C[1],C[2]), -k2*Cc(C[0],C[1],C[2])*C[2], k1*C[0]*C[1]/(1+km1/(k2*C[2]))]


#Résolution du système EDO ; t_eval est une option qui permet d'ajouter des points de temps --> lissage
res = solve_ivp(rhs, (0, 1), [Ca0, Cb0, Cd0, Ce0], t_eval = np.linspace(0, 1))


#Résolution de la concentration en C, qui dépend de Ca, Cb et Cd
ConcC = []
for i in range(len(res.t)):
    ConcC.append(Cc(res.y.T[i,0],res.y.T[i,1],res.y.T[i,2]))


#Légendes et axes
ax.set_title("pH = {}".format(pH_list[cpt]))
ax.set_xlabel('Temps (s)')
ax.set_ylabel('Concentration (mol/L)')
ax.plot(res[cpt].t, res[cpt].y.T)
```