# Centrale_2015_ecrit.py

```python

G, N = 1, 10

# IB
def smul(x, L):
    return [x*u for u in L]
# IC1
def vsum(L1, L2):
    return [L1[k] + L2[k] for k in range(len(L1))]
# IC2
def vdif(L1, L2):
    return [L1[k] - L2[k] for k in range(len(L1))]


# IIB2 l'équation est autonome, tmin et tmax inutiles
def euler(y0, z0, h, n, f):
    yi, zi = y0, z0
    y, z = [y0], [z0]
    for i in range(1, n):
        yi, zi = yi + h*zi, zi + h*f(yi)
        y.append(yi)
        z.append(zi)
    return y, z


# IIC1
def verlet(y0, z0, h, n, f):
    yi, zi = y0, z0
    y, z = [y0], [z0]
    for i in range(1, n):
        fi = f(yi)
        yi = yi + h*zi + (h**2*fi)/2
        zi = zi + h*(fi + f(yi))/2
        y.append(yi)
        z.append(zi)
    return y, z


# IIIA2
def force2(m1, p1, m2, p2):
    d2 = sum([x**2 for x in vdif(p1, p2)])
    cte = m1*m2*G / d2**1.5
    return smul(cte, vdif(p2, p1))


```

```
# IIIA3
def forceN(j, m, pos):
    force = 3*[0]
    for k in range(N):
        if k != j:
            force_k = force2(m[j], pos[j], m[k], pos[k])
            force = vsom(force, force_k)
    return force

# IIIB2
def pos_suiv(m, pos, vit, h):
    resultat = []
    for j in range(N):
        acceleration = smul(forceN(j, m, pos), 1/m[j])
        pos_s_j = vsom(vsom(pos[j], smul(h, vit[j])), smul(h**2/2, acceleration))
        resultat.append(pos_s_j)
    return resultat


import matplotlib.pyplot as plt
import numpy as np

def f(y):
    return - 4*(np.pi)**2*y

y0, z0 = 3, 0
n, h = 100, 1/33

y, z = euler(y0, z0, h, n, f)
plt.plot(y, z)

y, z = verlet(y0, z0, h, n, f)
plt.plot(y, z)
plt.show()

```