```
#1.
def Ep(x,y):
    return m*g*y+ 0.5*k1*((x**2+y**2)**0.5-l1)**2\
     +0.5*k2*(((L-x)**2+y**2)**0.5-l2)**2
#2.
def desc_grad(F,x0,y0,h,alpha,seuil):
    x=[x0]; y=[y0]; NG = 1+seuil # norme de G,
    #qu'on initialise à > seuil!!
    VF=[F(x0,y0)]; i=0
    while (NG>seuil) and (i<100):
        gx=F(x[i]+h,y[i])-F(x[i]-h,y[i])
        # on ne divise pas par 2h, inutile,
        # puisqu'on norme le vecteur ensuite
        gy=F(x[i],y[i]+h)-F(x[i],y[i]-h)
        NG= (gx**2+gy**2)**0.5
        x.append(x[i]-alpha*gx/NG)
        y.append(y[i]-alpha*gy/NG)
        i=i+1
        VF.append(F(x[i],y[i]))
    return [x,y,VF,i]
#3.
g=9.81; m=0.1/g; k1=2; l1=0.8; k2=1.2; l2,L=1,1
print(desc_grad(Ep,-0.02,-0.02,0.05,0.08,0.005))
#4.
def scal(L,M):
    return(L[0]*M[0]+L[1]*M[1])
#5.
def grad_conj(F,x0,y0,h,seuil):
    x=[x0]; y=[y0]; NG = 1+seuil
    PF = F(x0,y0); VF = [PF]; i=0
    gx=F(x[i]+h,y[i])-F(x[i]-h,y[i])
    gy=F(x[i],y[i]+h)-F(x[i],y[i]-h)
    g_vx=[gx,gy]
    d_vx=[gx,gy]
    while (NG>seuil) and (i<100):
        gx=F(x[i]+h,y[i])-F(x[i]-h,y[i])
        gy=F(x[i],y[i]+h)-F(x[i],y[i]-h)
        g=[gx,gy]
        #calcul de la descente optimisée d:
        Delta=[gx-g_vx[0],gy-g_vx[1]]
        c=scal(g,Delta)/scal(g_vx,g_vx)
        d=[gx-c*d_vx[0],gy-c*d_vx[1]]
        g_vx=g[:]
        d_vx=d[:]
        for k in range(0,501):#optimisation de alpha
            amin=k*0.1
            FMin=F(x[i]-amin*d[0],y[i]-amin*d[1])
            if FMin< PF:
                PF= FMin
                a=amin
```

```
        NG= (gx**2+gy**2)**0.5
        x.append(x[i]-a*d[0])
        y.append(y[i]-a*d[1])
        i=i+1
        VF.append(F(x[i],y[i]))
    return x,y,VF,i
D=grad_conj(Ep,-0.02,-0.02,0.05,0.005)
print('abscisses : ',D[0],'ordonnées : ',D[1],\
 'Energie potentielle :',D[2],'indice de fin :', D[3])
```