```
# -*- coding: utf-8 -*-

#####   EULER   ########
import matplotlib.pyplot as plt
from math import exp,cos,sqrt,atan,log10

def euler(F,y0,tmax,N):
    t=[0]
    y=[y0]
    h=tmax/N
    for k in range(N):
        t.append(k*h)
        y.append(y[k]+h*F(y[k],t[k]))

    return t,y

def F(y,t):
    return 10*(10-y)

t,u=euler(F,0,1,100)
usa=[10*(1-exp(-10*tt)) for tt in t ]

#plt.plot(t,u)
#plt.plot(t,usa)
plt.show()


def F2(y,t):
    if t<=0.05:
        return 10*(10-y)
    else:
        return 10*(-y)

t2,u2=euler(F2,0,2,100)
plt.clf()
#plt.plot(t2,u2)
#plt.show()


##### F.F.T. ##########
import numpy as np

def x(t):  ###créaction de x(t)=A sin(2 pi f t)
    f=1
    A=1
    return A*np.sin(2*np.pi*f*t)

### Echantillonnage
duree=3
Te=0.01
N=int(duree/Te) + 1
te=np.linspace(0,duree,N)
xe=x(te)
```

```
plt.clf()
#plt.scatter(te,xe,label="signal echantillonne")
#plt.grid()
#plt.show()


### calcul FFT D'UN SINUS

from numpy.fft import fft, fftfreq
X=fft(xe)
#freq=fftfreq(xe.size,d=Te)
N=len(xe)
freq=fftfreq(N,d=Te)

X_abs=np.abs(X[:N//2])
X_norm=X_abs*2.0/N
freq_pos=freq[:N//2]

plt.clf()
#plt.plot(freq_pos,X_norm, label="amplitude absolue")
plt.xlim(0,10)
plt.grid()
plt.xlabel(r"Freq(Hz)")
plt.ylabel(r"amplitude")
plt.title("Transformee de Fourier")
#plt.show()

#### FFT D'UNE GRANDEUR MESUREE

f=open("cren.csv","r") #Creer un fichier texte avec les donnees

po=[]     #creaction du point
for line in f:
    po+= line.split('\t')       #une histoire de \t et \n
f.close()

xx=[]   #creation des abscisses
yy=[]   #ordonnees
i=0
while i in range(len(po)-1):
    nx=float(po[i].replace(',','.') ) #si les decimaux sont ecrits
avec ,
                                      # au lieu de .
    xx+=[nx]
    ny=float(po[i+1].replace(',','.') )
    yy+=[ny]
    i+=2

t=xx
v=yy
plt.clf()
plt.plot(t,v)
plt.show()
Te=t[1]-t[0]
```

```
u=np.array(v)
U=fft(u)
freq=fftfreq(u.size,d=Te)

N=len(u)

U_abs=np.abs(U[:N//2])
U_norm=U_abs*2.0/N
freq_pos=freq[:N//2]

plt.clf()
plt.plot(freq_pos,U_norm, label="amplitude absolue")
plt.xlim(0,100)
plt.grid()
plt.xlabel(r"Freq(Hz)")
plt.ylabel(r"amplitude")
plt.title("Transformee de Fourier du créneau")
plt.show()


####### RECOMPOSITION DU CRENEAU #####

def trace(n,tmax,f0,N):
    h=tmax/N
    t=[k*h for k in range(N)]
    f=[(2*k+1)*f0 for k in range(n+1)]
    c=[]
    for i in range(N):
        s=0
        for j in range(n):
            s+=(-1)**j/(2*j+1)*cos(2*np.pi*f[j]*t[i])
        c.append(s)
    plt.clf()
    plt.plot(t,c)
    plt.show()

def cren(n,tmax,f0,N):
    h=tmax/N
    t=[k*h for k in range(N)]
    f=[(2*k+1)*f0 for k in range(n+1)]
    c=[]
    for i in range(N):
        s=0
        for j in range(n):
            s+=(-1)**j/(2*j+1)*cos(2*np.pi*f[j]*t[i])
        c.append(s)
    return t,c

#trace(5,10,1,1000)

tt,cc=cren(10,1,50,10000)

c=np.array(cc)
```

```
Te=tt[1]-tt[0]

C=fft(c)
N=len(c)
freq=fftfreq(N,d=Te)



C_abs=np.abs(C[:N//2])
C_norm=C_abs*2.0/N
freq_pos=freq[:N//2]

plt.clf()
#plt.plot(freq_pos,C_norm, label="amplitude absolue")
plt.xlim(0,200)
plt.grid()
plt.xlabel(r"Freq(Hz)")
plt.ylabel(r"amplitude")
plt.title("Transformee de Fourier")
#plt.show()


##### DIAGRAMME DE BODE

def gdb(f,fc):
    return 20*log10(1/sqrt(1+f**2/fc**2))

def arg(f,fc):
    return -atan(f/fc)
plt.clf()
freq=[k for k in range(10000)]
gain=[gdb(f,100) for f in freq]
phase=[arg(f,100) for f in freq]
#plt.plot(freq,gain)
plt.xscale("log")
#plt.show()
plt.plot(freq,phase)
plt.xscale("log")
#plt.show()

###### Creneau filtré

def cren(n,tmax,f0,N):
    h=tmax/N
    t=[k*h for k in range(N)]
    f=[(2*k+1)*f0 for k in range(n+1)]
    c=[]
    for i in range(N):
        s=0
        for j in range(n):
            s+=(-1)**j/(2*j+1)*cos(2*np.pi*f[j]*t[i])
        c.append(s)
    return t,c
```

```
t,c=cren(10,1,50,3000)

def crenf(t,c,gain,phi,n,tmax,f0,N):
    h=tmax/N
    t=[k*h for k in range(N)]
    f=[(2*k+1)*f0 for k in range(n+1)]
    c=[]
    for i in range(N):
        s=0
        for j in range(n):
            s+=10**(gain(f[j],100)/20)*(-1)**j/(2*j
+1)*cos(2*np.pi*f[j]*t[i]+arg(f[j],100))
        c.append(s)
    return t,c
tf,cf=crenf(t,c,gdb,arg,10,1,50,3000)

plt.clf()
#plt.plot(t,c)
plt.xlim(0,0.2)
#plt.plot(tf,cf)
plt.xlim(0,0.2)
#plt.show()

cren=np.array(cf)
Te=tf[1]-tf[0]

CF=fft(cren)
N=len(cren)
freq=fftfreq(N,d=Te)



CF_abs=np.abs(CF[:N//2])
CF_norm=CF_abs*2.0/N
freq_pos2=freq[:N//2]

plt.clf()
#plt.plot(freq_pos2,CF_norm, label="amplitude absolue")
plt.grid()
plt.xlim(0,400)
plt.xlabel(r"Freq(Hz)")
plt.ylabel(r"amplitude")
plt.title("Transformee de Fourier du signal filtré")
#plt.show()
#plt.plot(freq_pos,C_norm, label="amplitude absolue")
plt.xlim(0,400)
plt.grid()
plt.xlim(0,400)
plt.xlabel(r"Freq(Hz)")
plt.ylabel(r"amplitude")
plt.title("Transformee de Fourier du signal avant action du filtre")
#plt.show()

####EFFETS NON LINEAIRES ###
```

```
def x(t):  ###créaction de x(t)=A sin(2 pi f t)
    f=1
    A=1
    return A*np.sin(2*np.pi*f*t)


duree=3
Te=0.01
N=int(duree/Te) + 1
te=np.linspace(0,duree,N)
xes=[]
for t in te:
  if abs(x(t))<=0.5:
    xes.append(x(t))
  elif x(t)<-0.5:
    xes.append(-0.5)
  else:
    xes.append(0.5)


plt.clf()
plt.plot(te,xes)
plt.grid()
#plt.show()


Xs=fft(xes)
freq=fftfreq(len(xes),d=Te)
N=len(xes)
freq=fftfreq(N,d=Te)

Xs_abs=np.abs(Xs[:N//2])
Xs_norm=Xs_abs*2.0/N
freq_pos=freq[:N//2]

plt.clf()
plt.plot(freq_pos,Xs_norm, label="amplitude absolue")
plt.xlim(0,10)
plt.grid()
#plt.show()
```