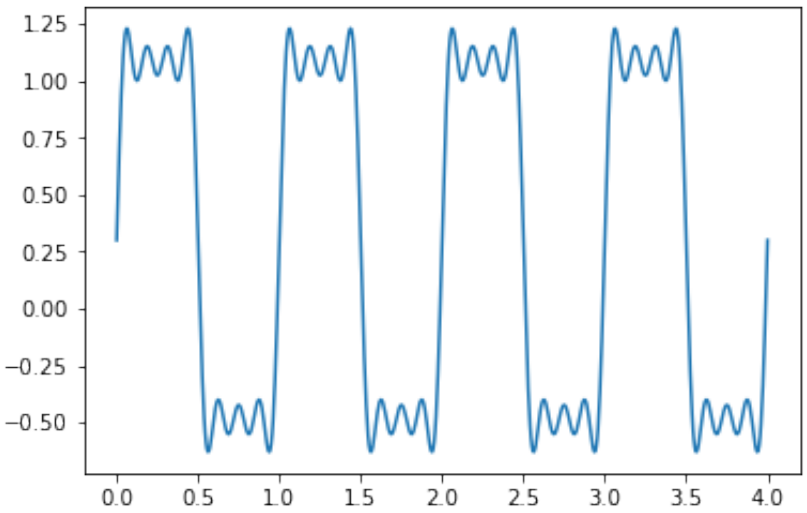
Eléments de traitement du signal

Chapitre 2 - Analyse spectrale

Il est important d'avoir une bonne connaissance des spectres des signaux usuels et de connaitre l'influence du filtrage et de l'échantillonnage sur le spectre des signaux.

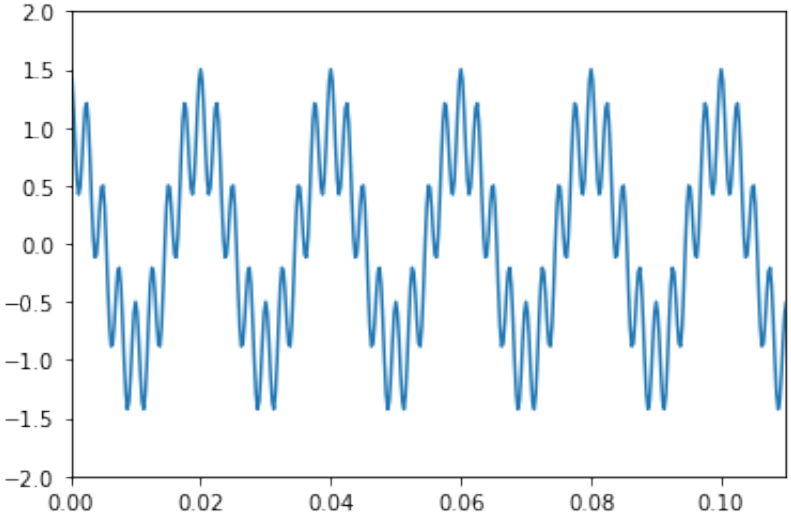
On a vu dans le TP 2 que le spectre d'un signal créneau est composé d'harmoniques impaires d'amplitude décroissant en 1/n. La présence de discontinuités se traduit par la présence d'harmoniques de rang élevé. La valeur moyenne du signal correspond à la composante du spectre à fréquence nulle.

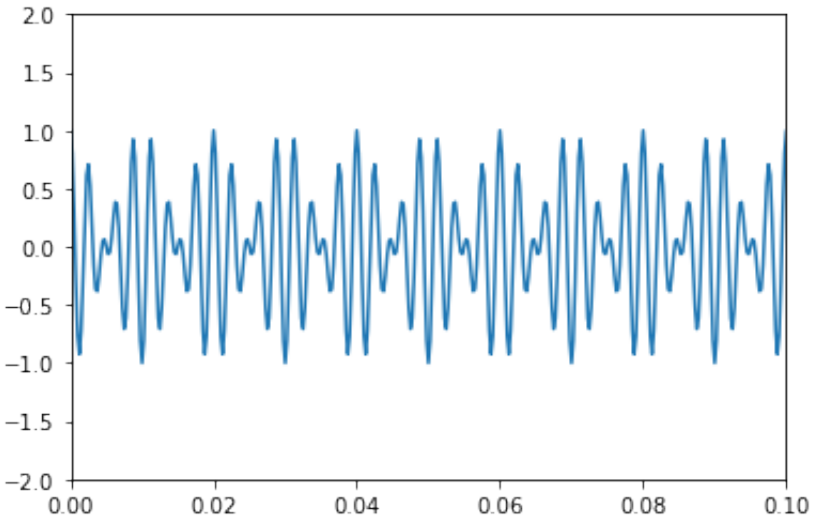
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

tc=np.linspace(0,4,700)
valmoy=0.3
creneau=valmoy+np.sin(2*np.pi*tc)+1/3*np.sin(2*np.pi*3*tc)+1/5*np.s
in(2*np.pi*5*tc)+1/7*np.sin(2*np.pi*7*tc)

plt.plot(tc,creneau)
plt.show()
```



On va maintenant créer deux signaux à partir de deux sinusoïdes de fréquences f1=50Hz et f2=400Hz en simulant une expérience dans laquelle l'échantillonnage se ferait à 4kHz.

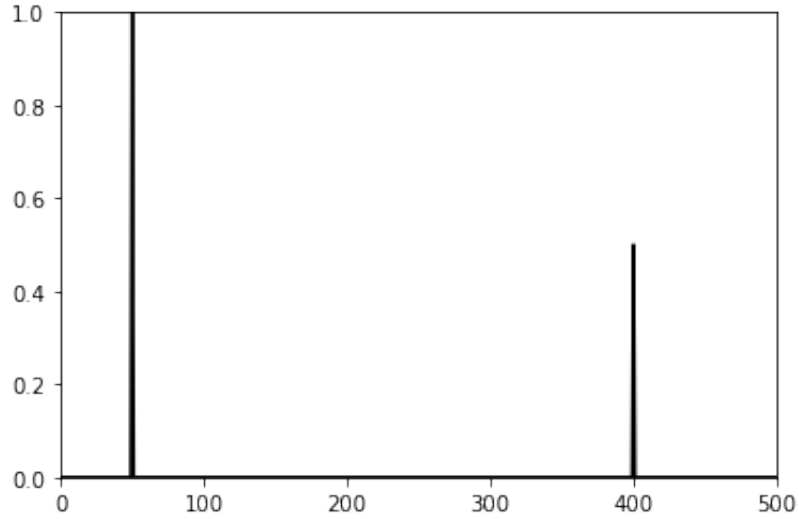
In [2]:
```
fe=4e3 #fréquence d'échantillonnage
f1=50 #fréquence 1
f2=400 #fréquence 2

Tacq=1 #durée de l'acquisition : 1s ici

Te=1/(fe)



# Création du premier signal : somme des deux sinusoïdes
t=np.arange(0,Tacq,Te)
N=len(t)
signal1=np.cos(2*np.pi*f1*t)+0.5*np.cos(2*np.pi *f2*t)


plt.axis([0,0.11,-2,2])
plt.plot(t,signal1)
plt.show()

# Le second signal est obtenu par produit.
signal2=np.cos(2*np.pi*f1*t)*np.cos(2*np.pi *f2*t)


plt.axis([0,0.1,-2,2])
plt.plot(t,signal2)
plt.show()
```





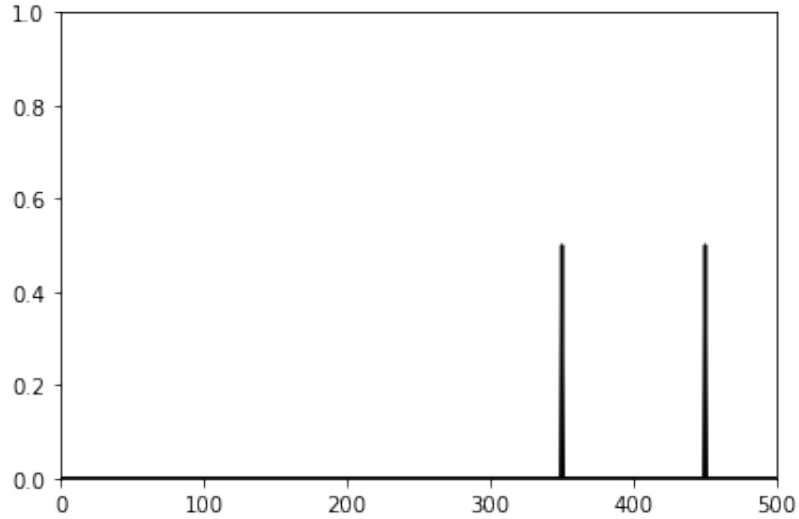
On peut représenter assez facilement le premier signal obtenu par somme : bruit haute fréquence de faible amplitude sur un signal haute fréquence ou signal haute fréquence dont la valeur moyenne fluctue suivant les amplitudes relatives des deux signaux. Le signal obtenu en multipliant les deux sinusoïdes est moins facile à interpréter : on peut alors calculer le spectre du signal pour une meilleure analyse.

In [4]:
```
fourier1 = np.fft.rfft(signal1)

freq = np.fft.rfftfreq(N, d=Te)

plt.plot(freq,np.abs(fourier1[0:len(freq)])*2/N,'k-')
plt.axis([0,500,0,1])
plt.show()
```



On observe bien un pic à f1 et un pic à f2 dans le cas du signa1 obtenu en sommant les deux signaux.

In [5]:
```
fourier2 = np.fft.rfft(signal2)

freq = np.fft.rfftfreq(N, d=Te)

plt.plot(freq,np.abs(fourier2[0:len(freq)])*2/N,'k-')
plt.axis([0,500,0,1])
plt.show()
```



Dans le cas du signal2 obtenu en multipliant les deux sinusoïdes on obtient un pic à f2-f1 et un pic à f2+f1.

$$\cos(p).\cos(q) = \frac{1}{2}(\cos(p+q) + \cos(p-q))$$

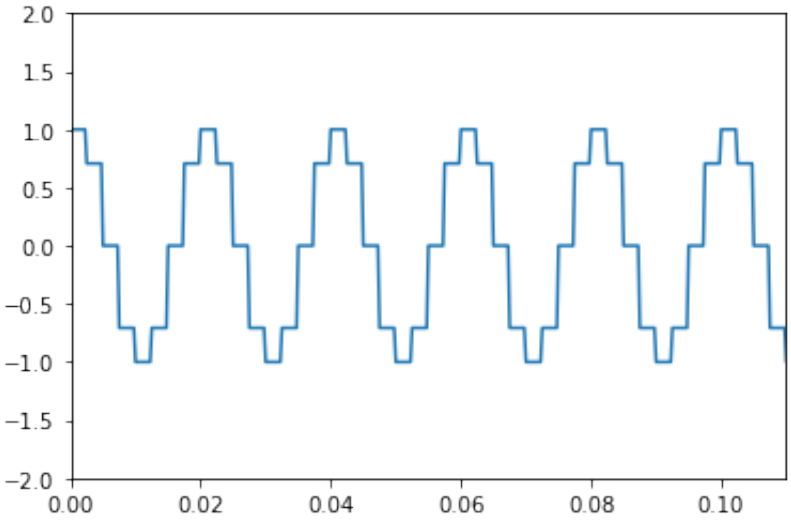
Dans le TP3, on a pu observer l'effet de l'échantillonnage à une fréquence fe sur le spectre d'un signal sinusoïdal de fréquence f1 : en plus du pic à f1, on observe des pics à fe-f1 et fe+f1 ; 2fe-f1 et 2fe-f1 ...
On en a déduit le critère de Shannon Nyquist :

$$fe > 2f1$$

pour éviter un repliement de spectre.

On peut simuler un signal issu d'un échantillonneur-bloqueur.

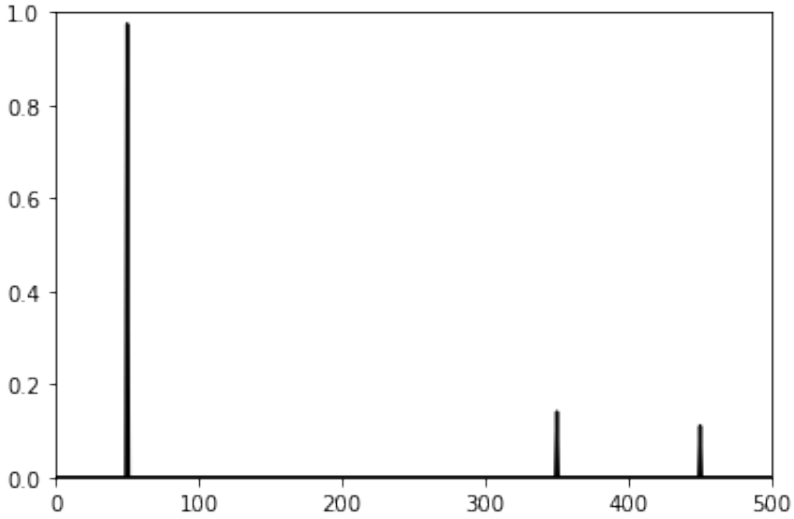
In [12]:
```
n=10
p=N//n
signal3=np.zeros(N)
for i in range (n):
    for k in range (p):
        signal3[k*n+i]=np.cos(2*np.pi*f1*t[n*k])
plt.plot(t,signal3)
plt.axis([0,0.11,-2,2])
plt.show()
print(fe/n) #il s'agit de la fréquence d'échantillonnage simulée
```




```
400.0
```

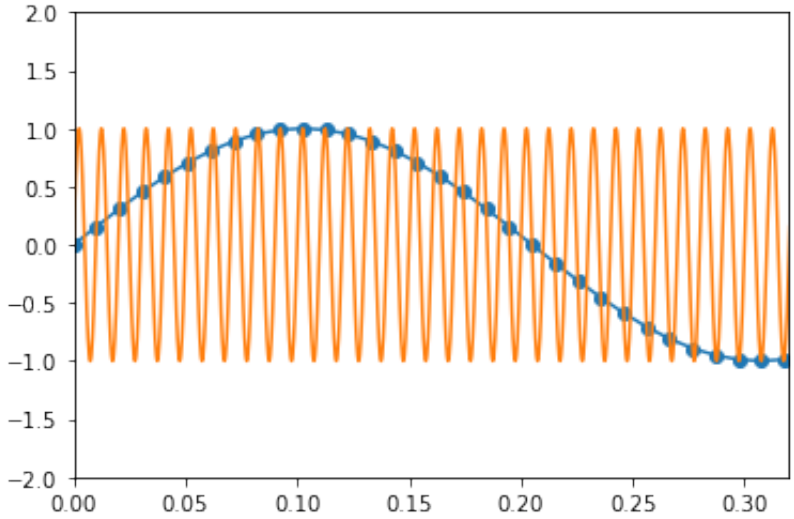
On trace ensuite le spectre de ce signal.

In [11]:
```
fourier3 = np.fft.rfft(signal3)

freq = np.fft.rfftfreq(N, d=Te)

plt.plot(freq,np.abs(fourier3[0:len(freq)])*2/N,'k-')
plt.axis([0,500,0,1])
plt.show()
```



On obtient bien un pic à fe/n -f1 et fe/n+f1 On peut aussi simuler l'effet d'un mauvais échantillonnage.

In [31]:
```
n=41
p=N//n
signal4=np.zeros(p)
t4=np.zeros(p)
for k in range (p):
    signal4[k]=np.sin(2*np.pi*100*t[n*k])
    t4[k]=t[n*k]
plt.plot(t4,signal4,'-o')
plt.plot(t,np.sin(2*np.pi*100*t))
plt.plot()
plt.axis([0,0.32,-2,2])
plt.show()
```



In [ ]: