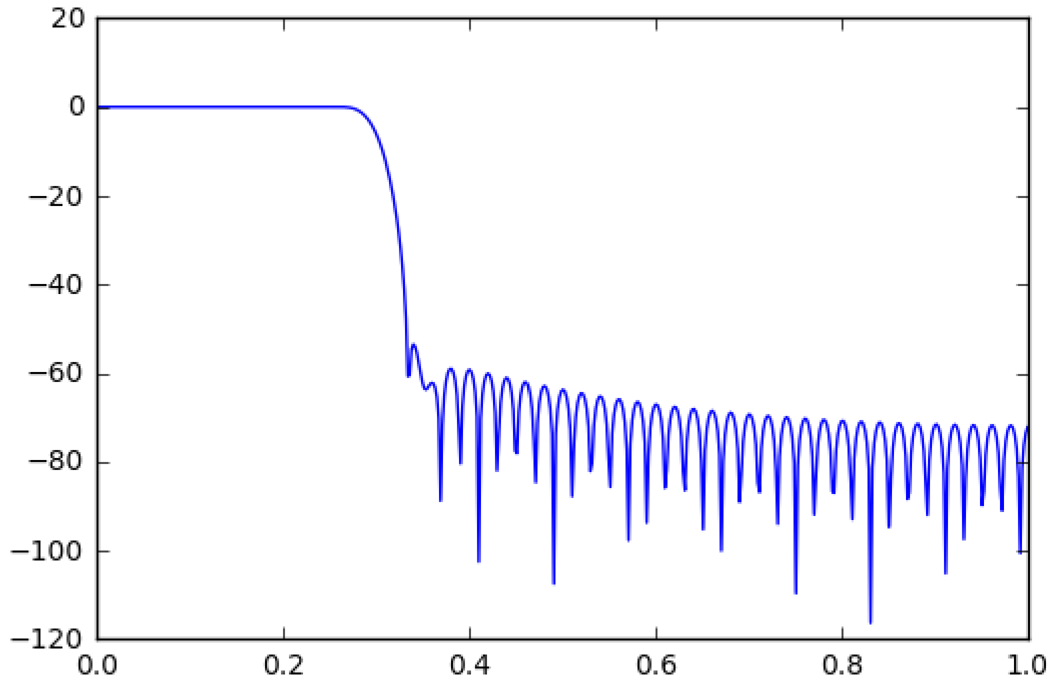
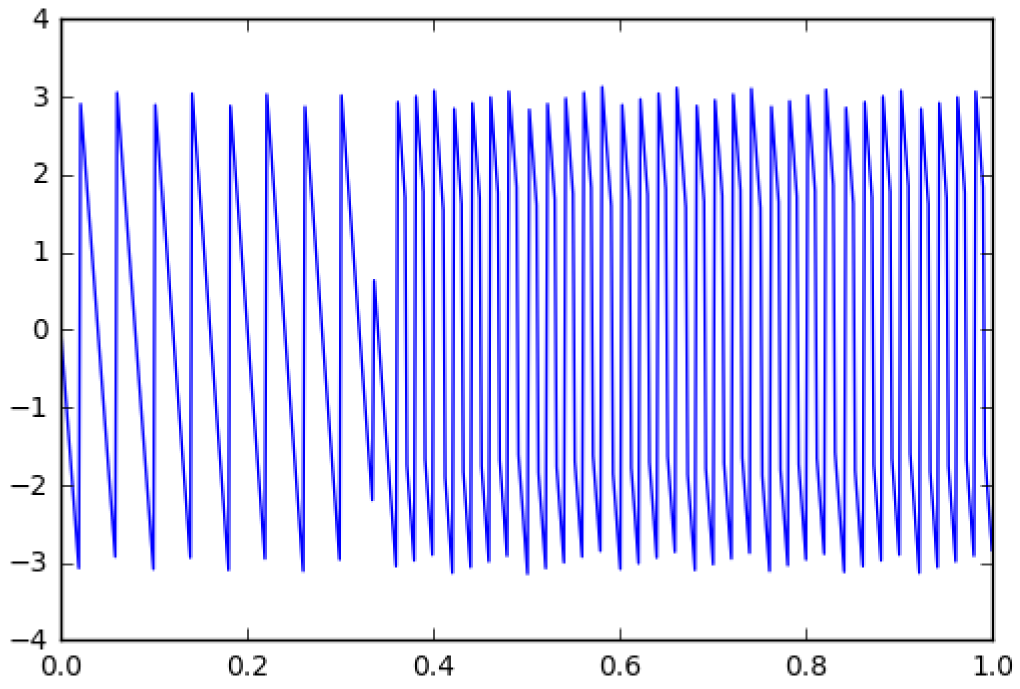
Answer-1

Q.1.(a)Determine and plot frequecy response of FIR Filters using Hamming windowing function. The details about FIR Filter Design are given:https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.firwin.html#scipy.signal.firwin (https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.firwin.html#scipy.signal.firwin)
Important window functions: hamming,hann, blackman, triang, bartlett, flattop, parzen, bohman, blackmanharris, nuttall, barthann, kaiser (needs beta), gaussian (needs standard deviation), general_gaussian (needs power, width), slepian (needs width), dpss (needs normalized half-bandwidth), chebwin (needs attenuation), exponential (needs decay scale), tukey (needs taper fraction)
Python Function example: signal.get_window('triang', 7)
(b)Repeat part(a) for hann, blackman, and bartlett windowing functions

In [74]:

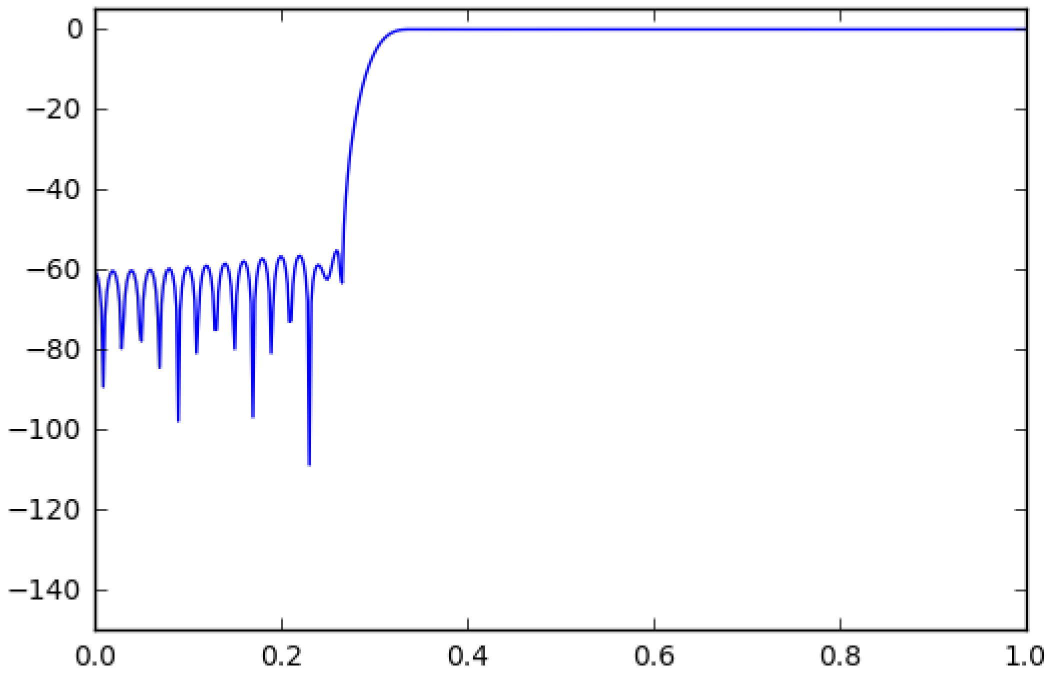
```
import numpy as np
import scipy.signal as sig
import matplotlib.pyplot as plt
a=1
n=101
b = sig.firwin(n, cutoff = 0.3, pass_zero=True, window = "hamming")
w,h = sig.freqz(b,a)
h_dB = 20 * np.log10 (abs(h))
angles = np.angle(h)
#plt.ylim(-150, 5)
plt.plot(w/max(w),h_dB)
plt.show()
plt.plot(w/max(w),angles )
plt.show()
```

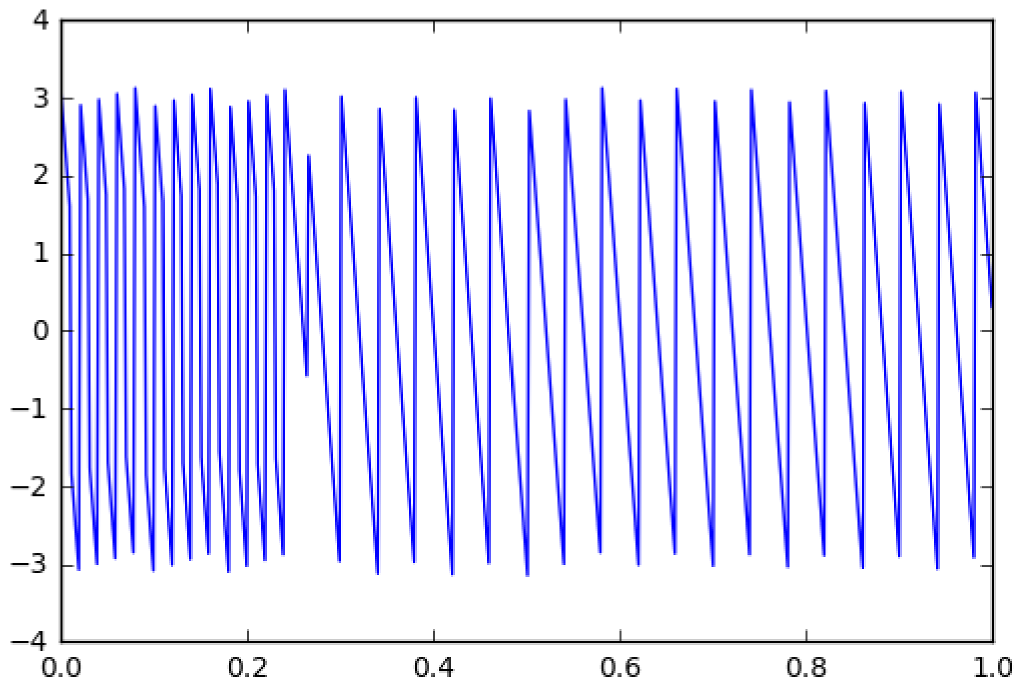




In [75]:

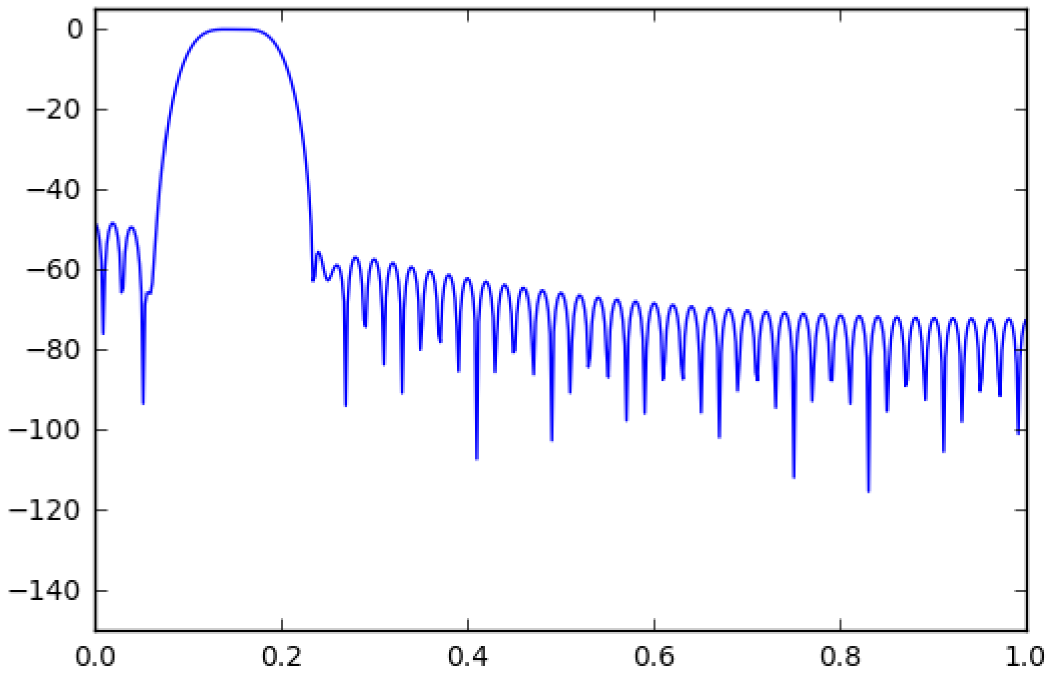
```
a=1
n = 101
b = sig.firwin(n,0.3, pass_zero=False, window = "hamming")
w,h = sig.freqz(b,a)
h_dB = 20 * np.log10 (abs(h))
angles = np.angle(h)
plt.ylim(-150, 5)
plt.plot(w/max(w),h_dB)
plt.show()
plt.plot(w/max(w),angles )
plt.show()
```

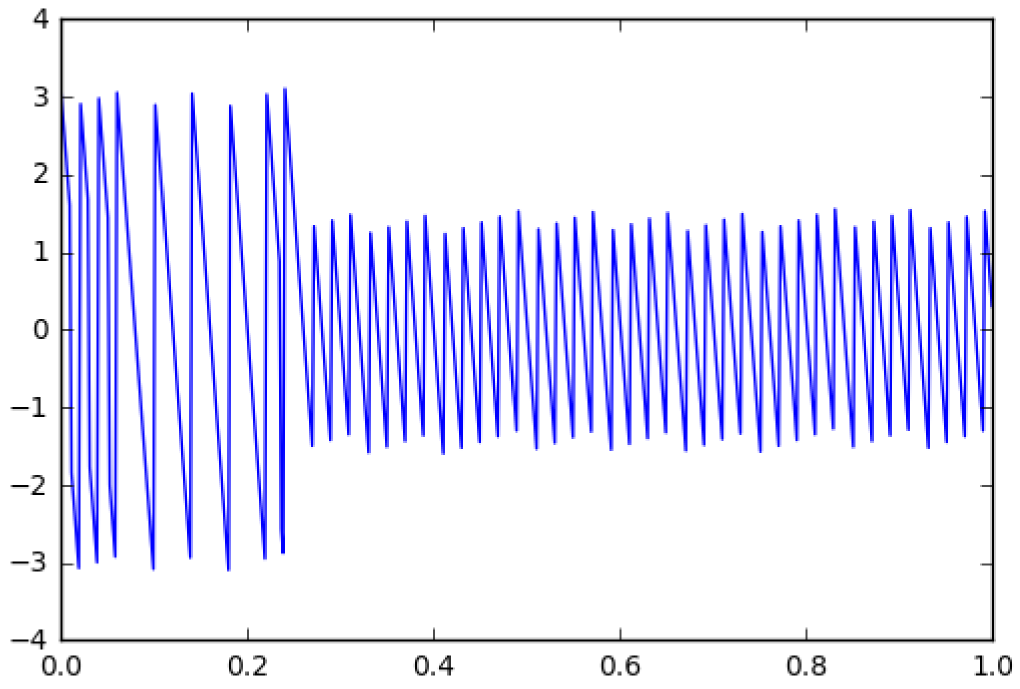




In [5]:

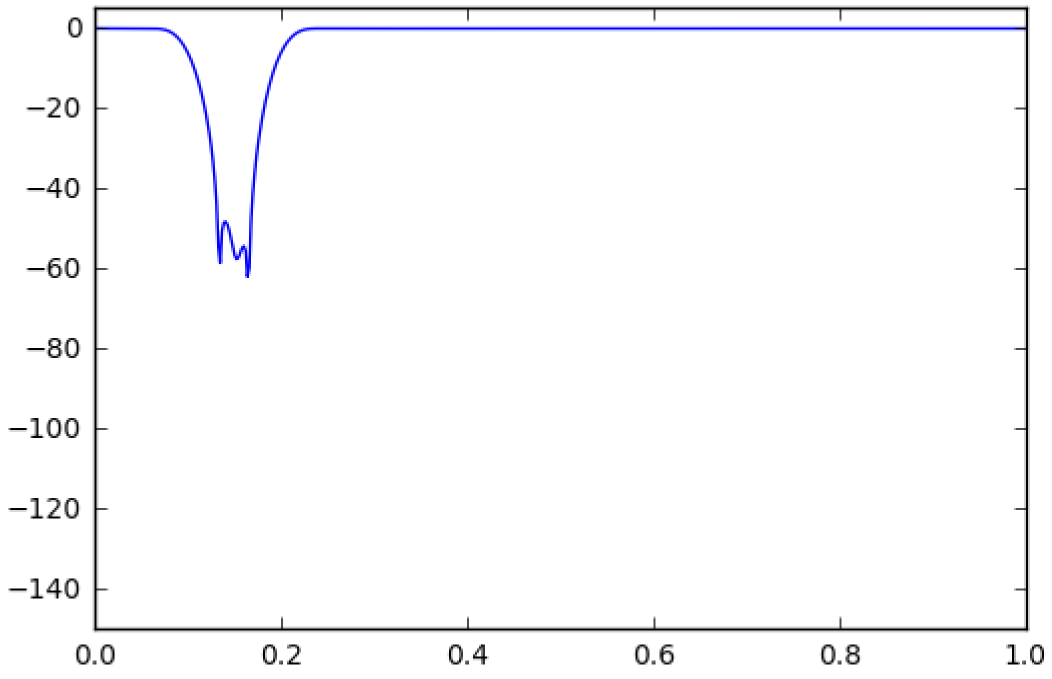
```
a=1
n = 101
f1, f2 = 0.1, 0.2
b = sig.firwin(n,[0.1, 0.2], pass_zero=False,window = "hamming")
w,h = sig.freqz(b,a)
h_dB = 20 * np.log10 (abs(h))
angles = np.angle(h)
plt.ylim(-150, 5)
plt.plot(w/max(w),h_dB)
plt.show()
plt.plot(w/max(w),angles )
plt.show()
```

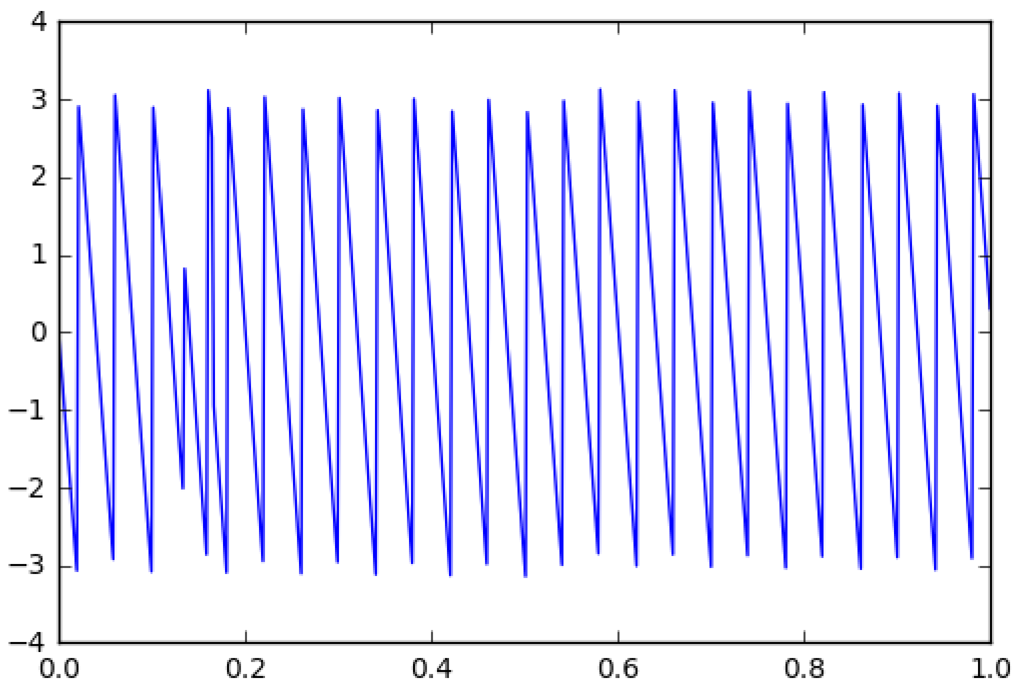




In [6]:

```
a=1
n = 101
f1, f2 = 0.1, 0.2
b = sig.firwin(n,[0.1, 0.2],window = "hamming")
w,h = sig.freqz(b,a)
h_dB = 20 * np.log10 (abs(h))
angles = np.angle(h)
plt.ylim(-150, 5)
plt.plot(w/max(w),h_dB)
plt.show()
plt.plot(w/max(w),angles )
plt.show()
```





Answer-2

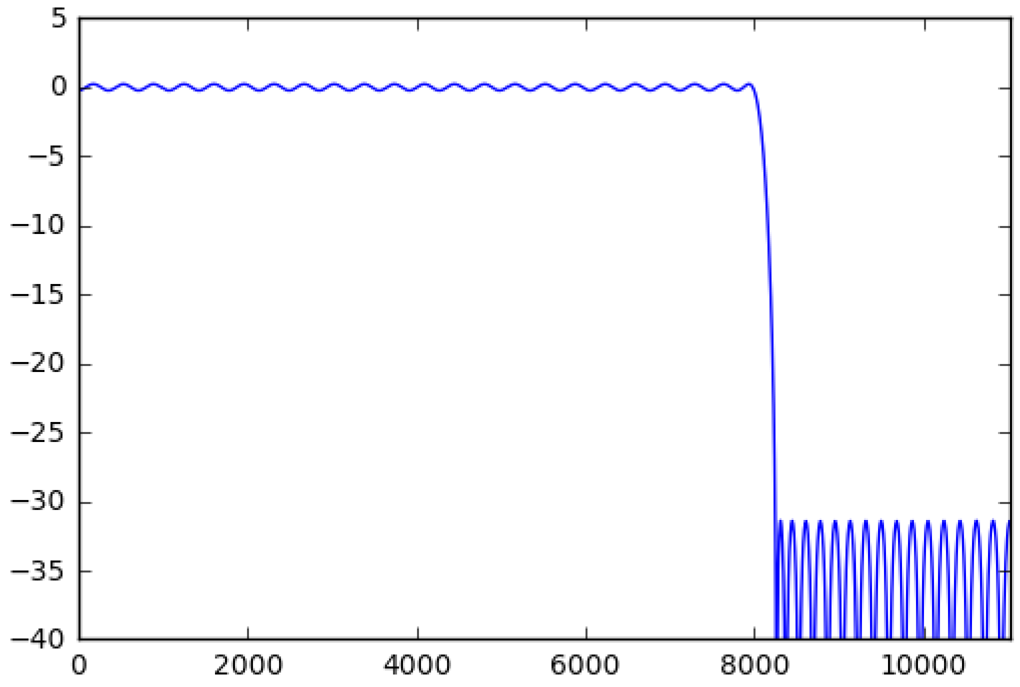
Q.2. Plot the frequecy resposne of LPF, HPF, BPF,and BSF FIR Filters with the following design parameters
(a)LPF and HPF
fs = 22050.0 # Sample rate, Hz
cutoff = 8000.0 # Desired cutoff frequency, Hz
trans_width = 250 # Width of transition from pass band to stop band, Hz
numtaps = 125 # Size of the FIR filter.

(b)BPF and BSF
fs = 22050.0 # Sample rate, Hz
band = [2000, 5000] # Desired stop band, Hz
trans_width = 250 # Width of transition from pass band to stop band, Hz
numtaps = 125 # Size of the FIR filter.

The details about Python function {scipy.signal.remez} are given:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.remez.html#scipy.signal.remez
(https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.remez.html#scipy.signal.remez).
{tabs=scipy.signal.remez(numtaps, bands, desired, weight=None, Hz=None, type='bandpass', maxiter=25, grid_density=16, fs=None)[source]}

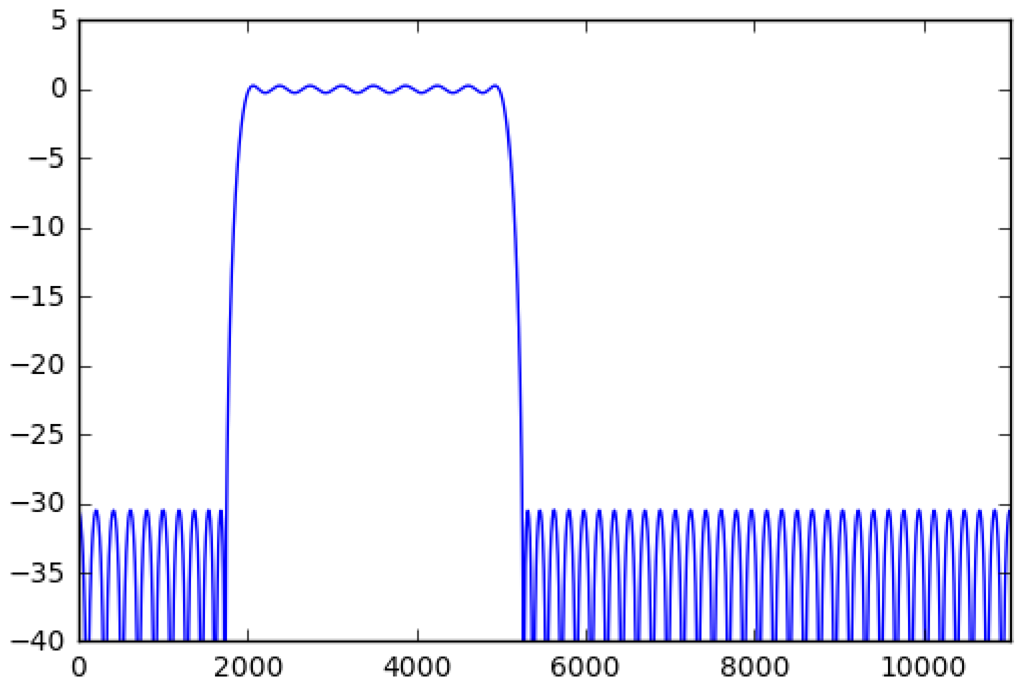
In [76]:

```
import numpy as np
import scipy.signal as sig
import matplotlib.pyplot as plt
fs = 22050.0       # Sample rate, Hz
cutoff = 8000.0    # Desired cutoff frequency, Hz
trans_width = 250  # Width of transition from pass band to stop band, Hz
numtaps = 125      # Size of the FIR filter.
taps = sig.remez(numtaps,[0, cutoff, cutoff + trans_width, 0.5*fs],[1, 0], Hz=fs)
w, h = sig.freqz(taps, [1], worN=2000)
plt.plot(0.5*fs*w/np.pi, 20*np.log10(np.abs(h)))
plt.ylim(-40, 5)
plt.xlim(0, 0.5*fs)
plt.show()
```



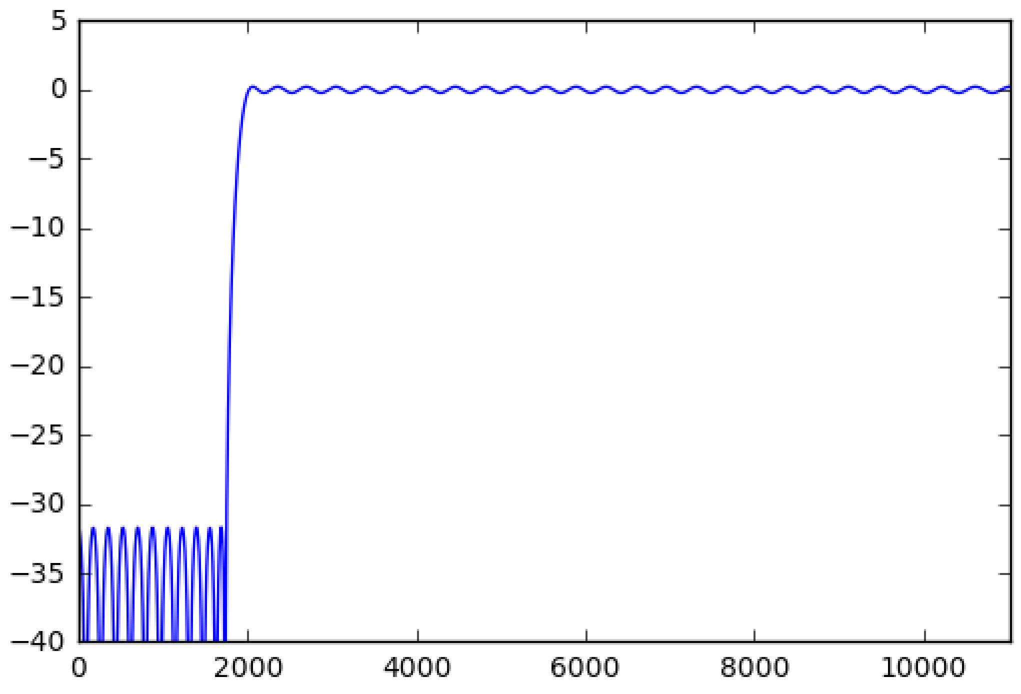
In [77]:

```
# Band-pass filter design parameters
fs = 22050.0         # Sample rate, Hz
band = [2000, 5000]  # Desired pass band, Hz
trans_width = 250    # Width of transition from pass band to stop band, Hz
numtaps = 125        # Size of the FIR filter.

edges = [0, band[0] - trans_width, band[0], band[1],band[1] + trans_width, 0.5*fs]
taps = sig.remez(numtaps, edges, [0, 1, 0], Hz=fs)
w, h = sig.freqz(taps, [1], worN=2000)
plt.plot(0.5*fs*w/np.pi, 20*np.log10(np.abs(h)))
plt.ylim(-40, 5)
plt.xlim(0, 0.5*fs)
plt.show()
```



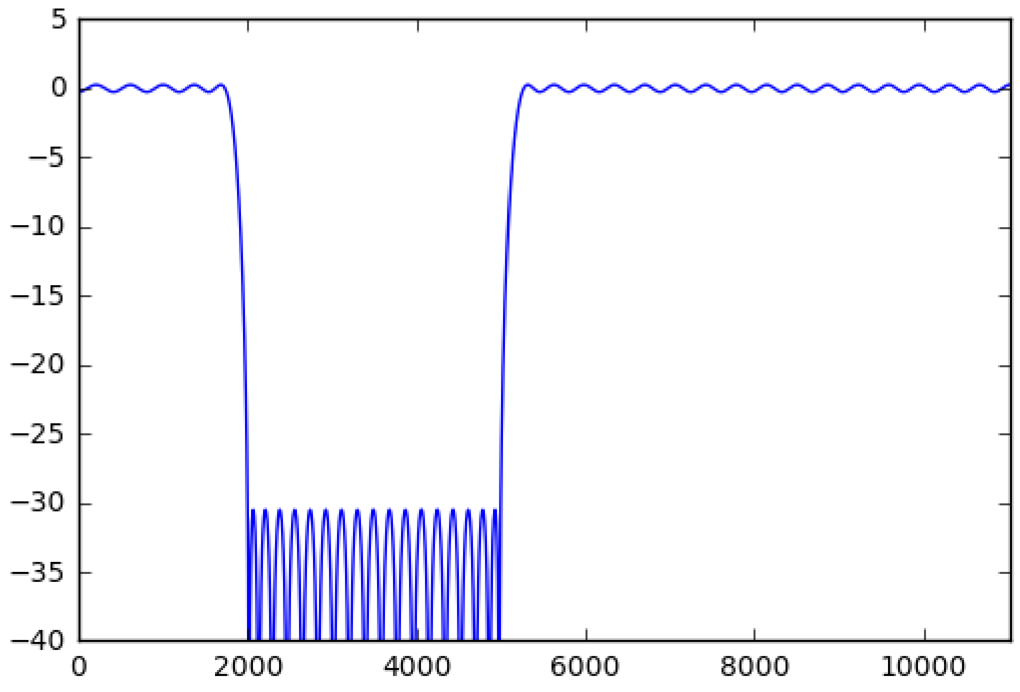
In [10]:

```
# High-pass filter design parameters
fs = 22050.0       # Sample rate, Hz
cutoff = 2000.0    # Desired cutoff frequency, Hz
trans_width = 250  # Width of transition from pass band to stop band, Hz
numtaps = 125      # Size of the FIR filter.

taps = sig.remez(numtaps, [0, cutoff - trans_width, cutoff, 0.5*fs],[0, 1], Hz=fs)
w, h = sig.freqz(taps, [1], worN=2000)
plt.plot(0.5*fs*w/np.pi, 20*np.log10(np.abs(h)))
plt.ylim(-40, 5)
plt.xlim(0, 0.5*fs)
plt.show()
```



In [78]:

```
# Band-stop filter design parameters
fs = 22050.0         # Sample rate, Hz
band = [2000, 5000]  # Desired stop band, Hz
trans_width = 250    # Width of transition from pass band to stop band, Hz
numtaps = 125        # Size of the FIR filter.

edges = [0, band[0] - trans_width,band[0], band[1],band[1] + trans_width, 0.5*fs]
taps = sig.remez(numtaps, edges, [1, 0, 1], Hz=fs)
w, h = sig.freqz(taps, [1], worN=2000)
plt.plot(0.5*fs*w/np.pi, 20*np.log10(np.abs(h)))
plt.ylim(-40, 5)
plt.xlim(0, 0.5*fs)
plt.show()
```

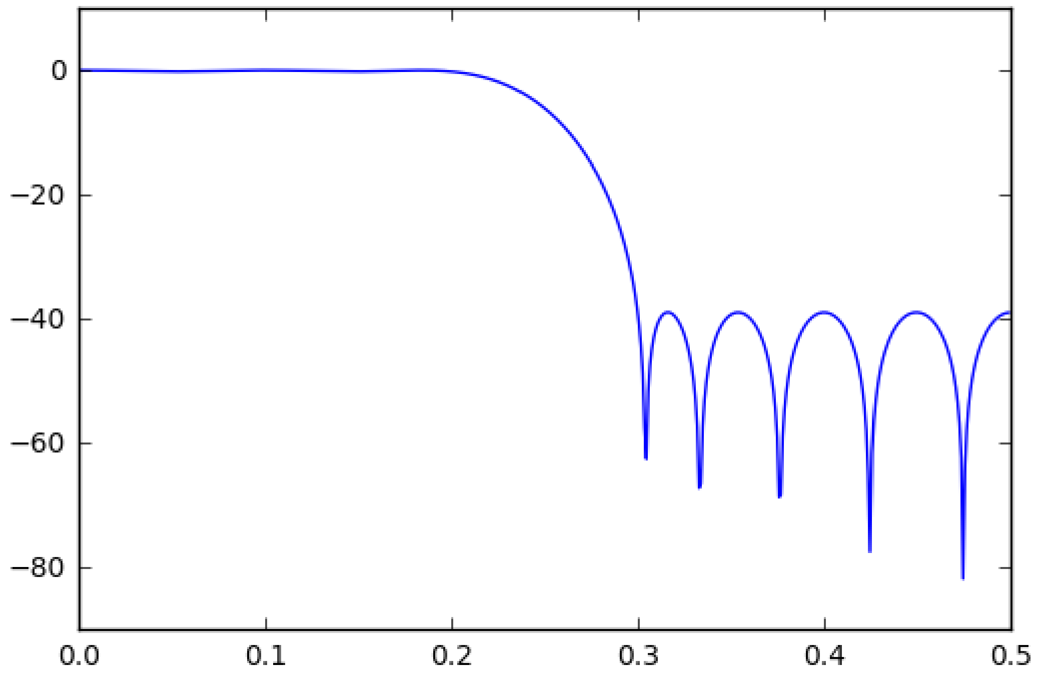


Answer-3

Q.3. (a) For the Filter coefficients, we need to design a Low FIR filter that will suppress the 200Hz component by at least 30dB while leaving the 100Hz component alone. The digiatl low FIR filter with design specifications: passband between 0 to 200 and stop band from 300 to 500 (half the sampling frequency of 1000Hz).
Python function: taps=scipy.signal.remez(numtaps, bands, desired, weight=None, Hz=None, type='bandpass', maxiter=25, grid_density=16, fs=None)
(b) Determine and plot the filtered output signal using FIR Remez Algorithm with design specifications mentioned in part(a) for the following signal and also compare between noisy signal and filted output signal: $x(n) = sin(0.2\pi n) + sin(1.2\pi n) + w[n]; 0 \leq n \leq 40$; Where $w(n)$ is ramdom noise with noise amplitude 20.

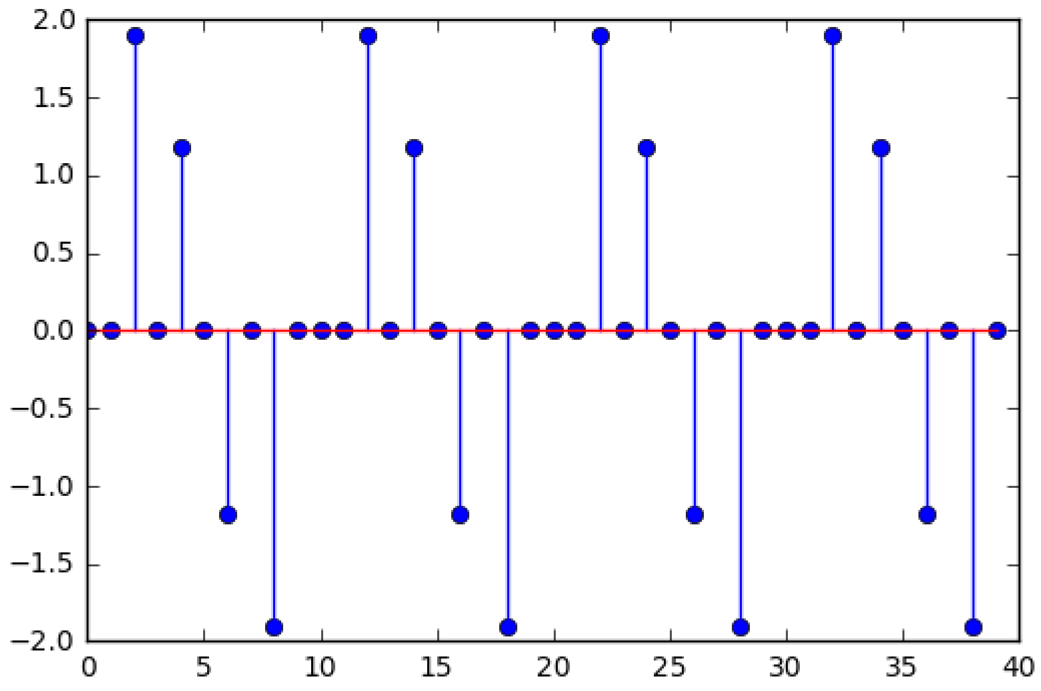
In [79]:

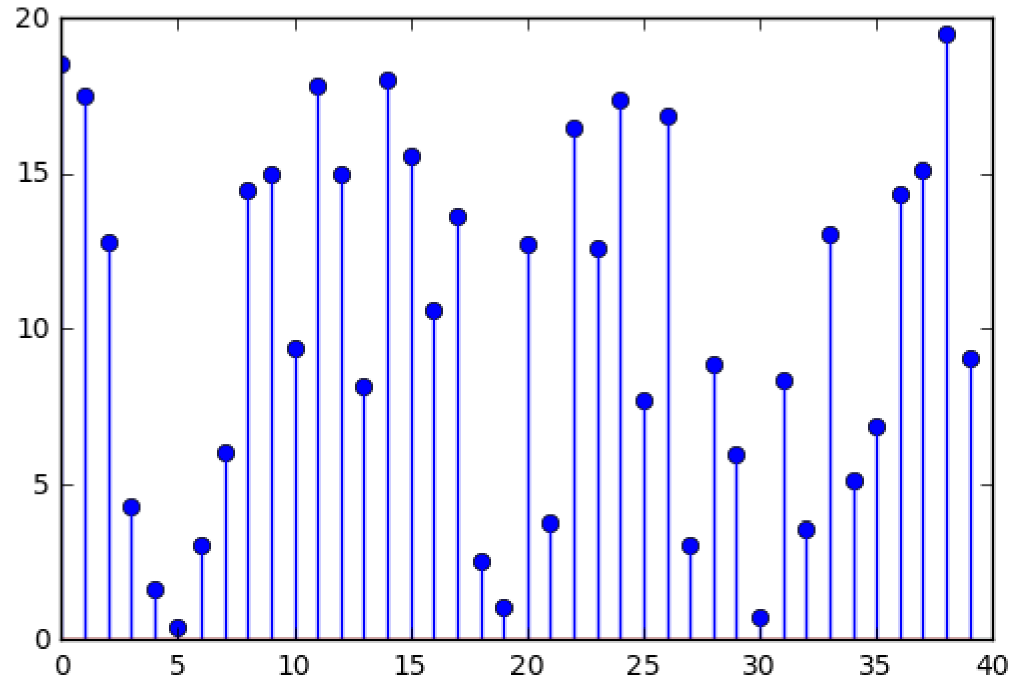
```
import numpy as np
import scipy.signal as sig
import matplotlib.pyplot as plt
#(a) Low pass filter
lpf =sig.remez(21, [0, 0.2, 0.3, 0.5], [1.0, 0.0])

w, h = sig.freqz(lpf)
plt.plot(w/(2*np.pi), 20*np.log10(abs(h)))
plt.show()
```



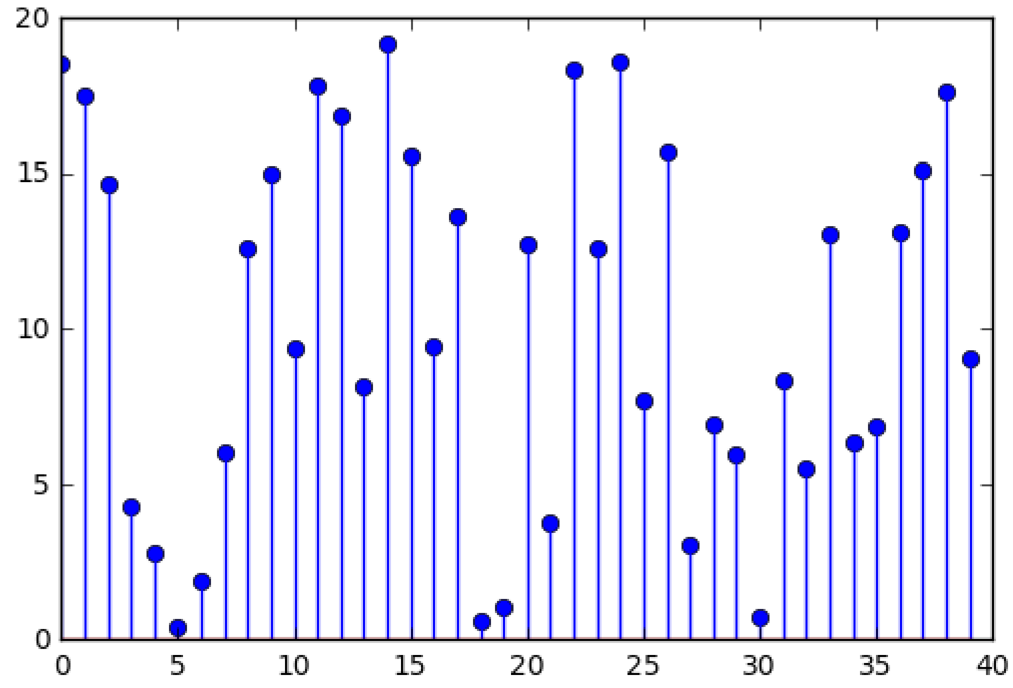
In [81]:

```
# (b) Application as Filter of noisy signal
# Signal
n=range(0,40)
k=np.array(n)
s1 = np.sin(2*np.pi*0.10*k)+np.sin(2*np.pi*0.60*k)
plt.stem(s1)
plt.show()
# Noise Signal
w=20*np.random.rand(len(n))
plt.stem(w)
plt.show()
# Noisy Signal
s=s1+w
plt.stem(s)
plt.show()
```





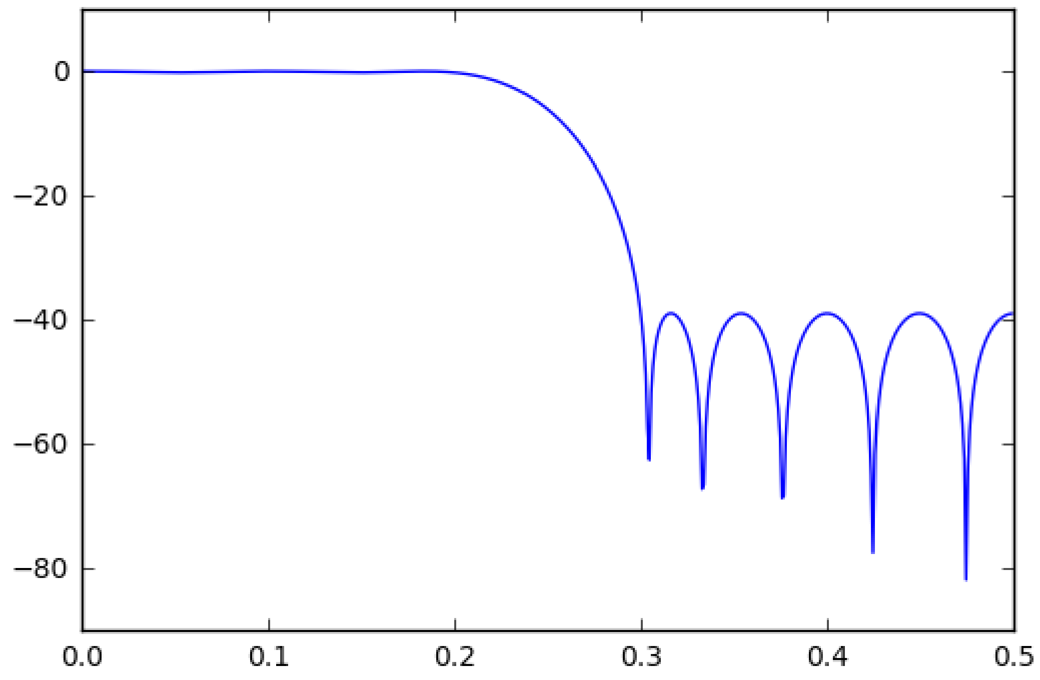
0
5
10
15
20
25
30
35
40



0
5
10
15
20
25
30
35
40

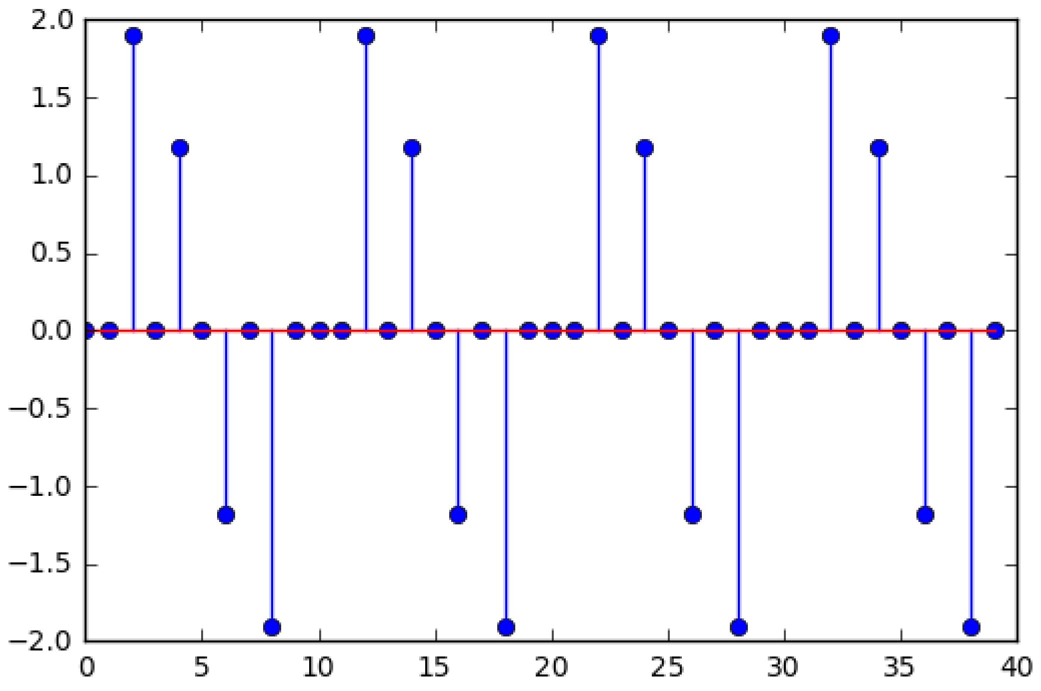
In [25]:

```
import numpy as np
import scipy.signal as sig
import matplotlib.pyplot as plt
#(a) Low pass filter
lpf =sig.remez(21, [0, 0.2, 0.3, 0.5], [1.0, 0.0])

w, h = sig.freqz(lpf)
plt.plot(w/(2*np.pi), 20*np.log10(abs(h)))
plt.show()
```



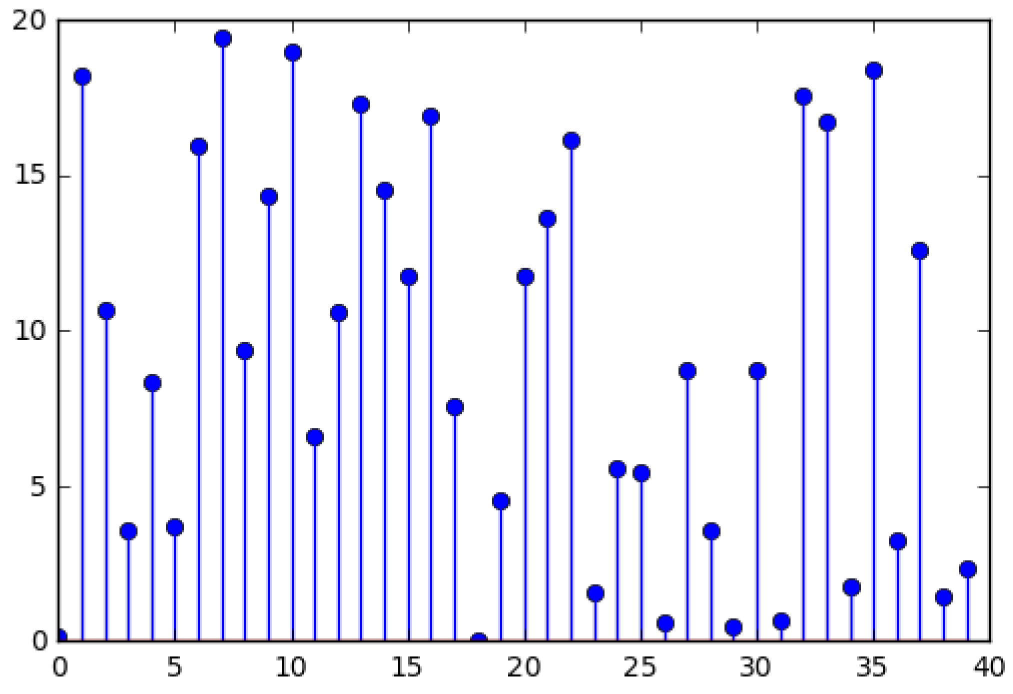
In [26]:

```
# (b) Application as Filter of noisy signal
# Signal
n=range(0,40)
k=np.array(n)
s1 = np.sin(2*np.pi*0.10*k)+np.sin(2*np.pi*0.60*k)
plt.stem(s1)
plt.show()
```



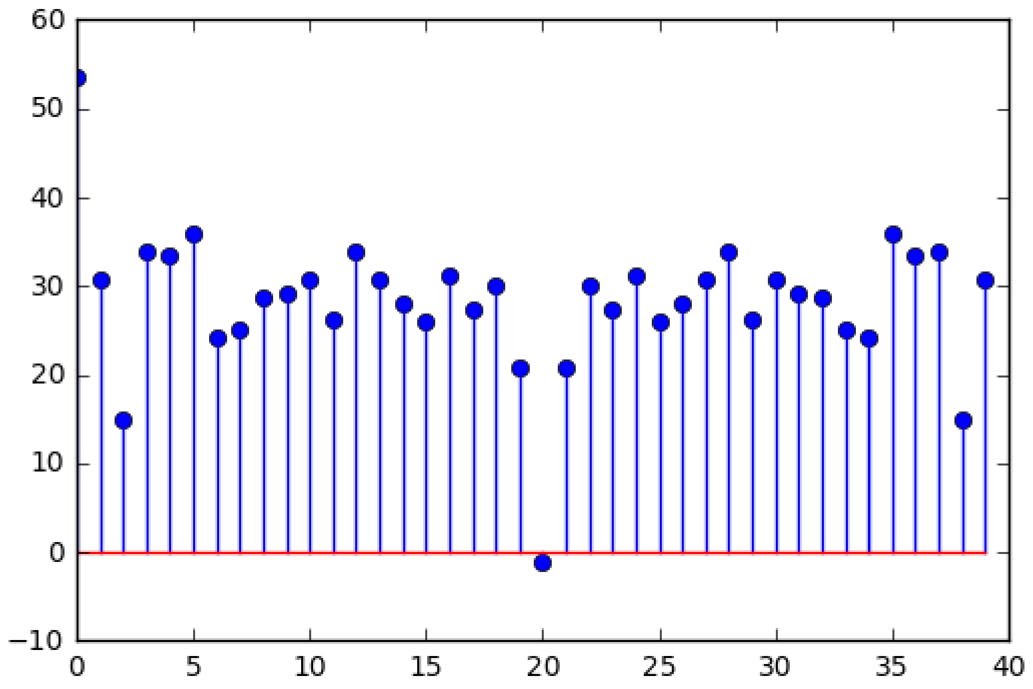
In [27]:

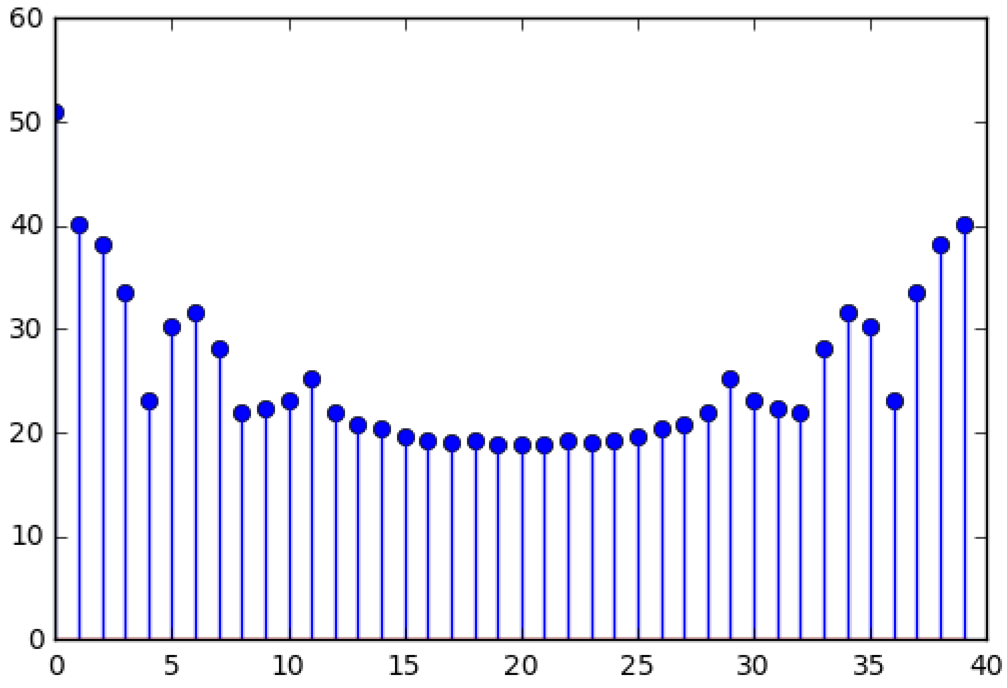
```
# Noise Signal
w=20*np.random.rand(len(n))
plt.stem(w)
plt.show()
```



In [29]:

```
from scipy.signal import lfilter
sout = lfilter(lpf, 1, s)
plt.stem(20*np.log10(abs(np.fft.fft(s))))
plt.show()
plt.stem(20*np.log10(abs(np.fft.fft(sout))))
plt.show()
```



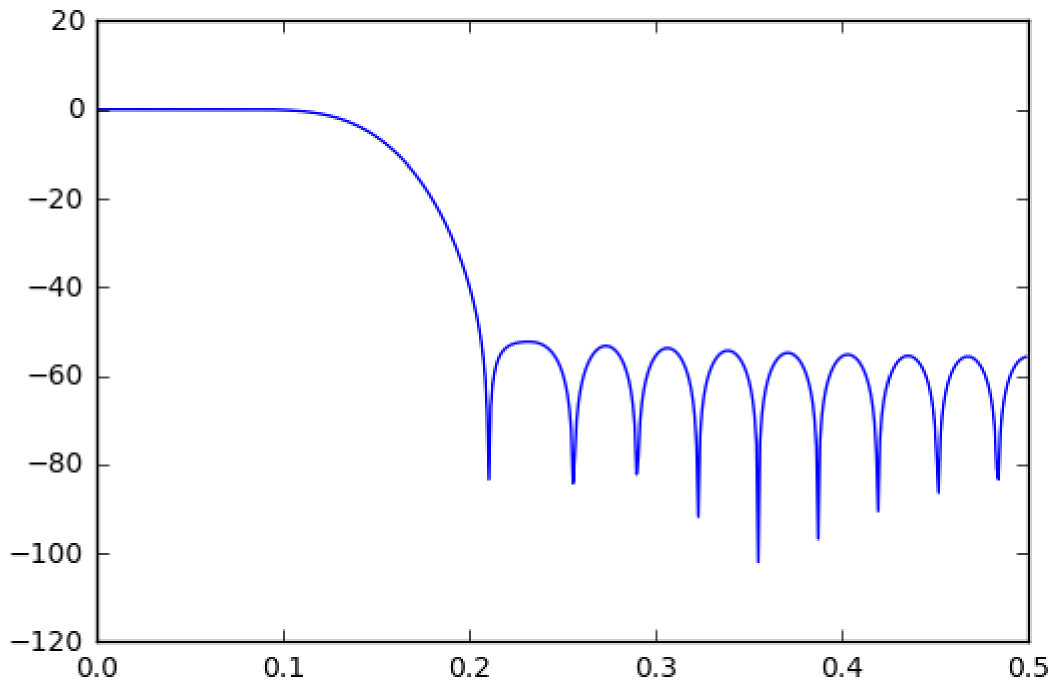


answer-4

Q.4.(a) Determine and plot the filtered output signal by FIR Hamming window function for the following signal and also compare between noisy signal and filted output signal:
$x(n) = sin(0.2\pi n) + sin(1.2\pi n) + w[n]; 0 \leq n \leq 60;$;Where $w(n)$ is ramdom noise with noise amplitude 20.
(b)Repeat part(a) for hann, blackman, and bartlett windowing functions

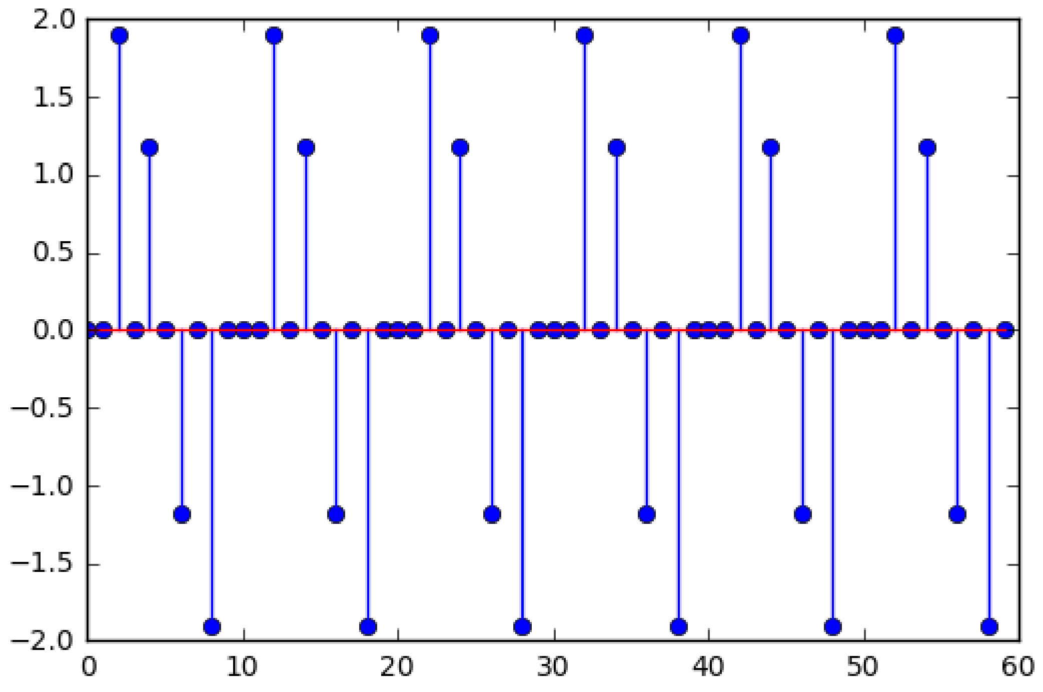
In [39]:

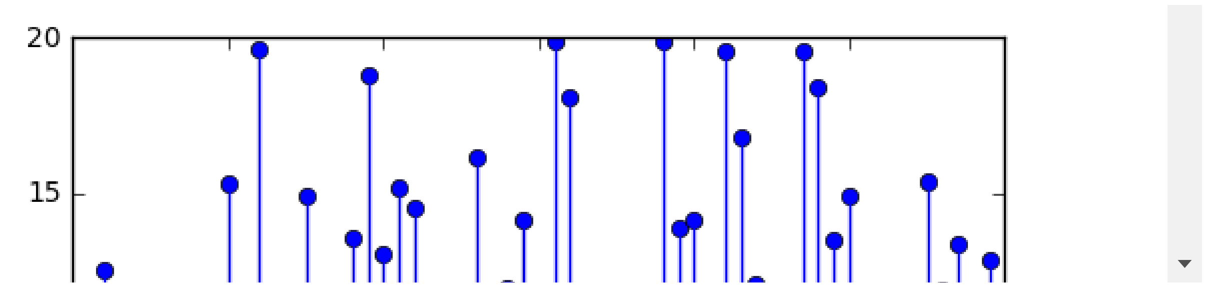
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

#Lowpass FIR filter
N = 31
b = sig.firwin(N, cutoff = 0.3, pass_zero=True, window = "hamming")
#Frequency response
w, h = sig.freqz(b)
plt.plot(w/(2*np.pi), 20*np.log10(abs(h)))
plt.show()
```

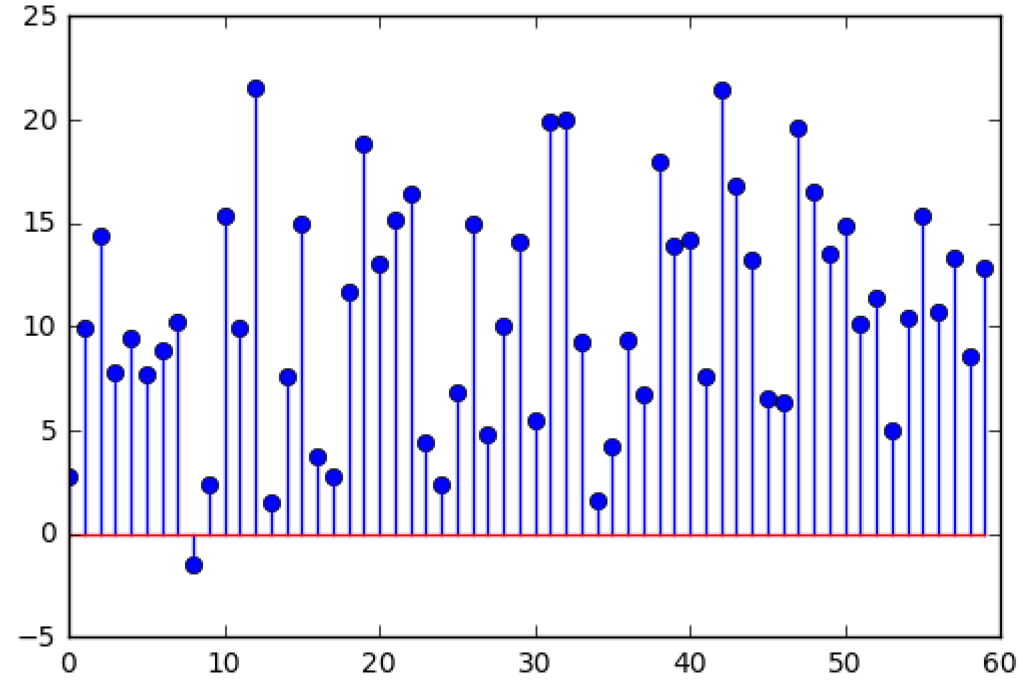


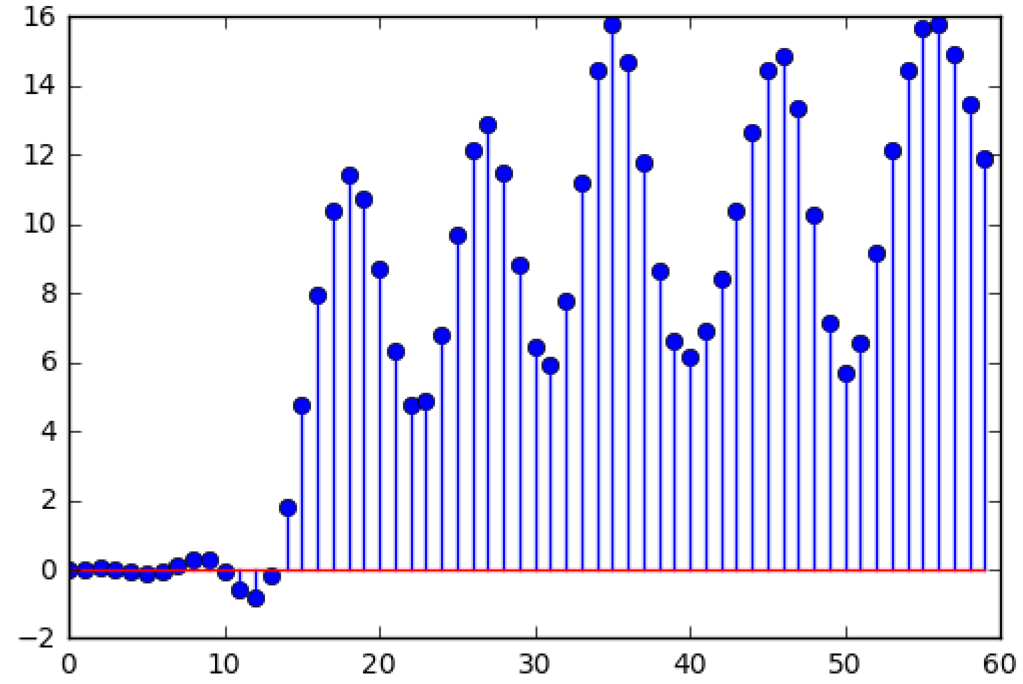
In [82]:

```
N = 31
b = sig.firwin(N, cutoff = 0.3, pass_zero=True, window = "hamming")
# Signal
n=range(0,60)
k=np.array(n)
s1 = np.sin(2*np.pi*0.10*k)+np.sin(2*np.pi*0.60*k)
plt.stem(s1)
plt.show()
# Noise Signal
# Noise Signal
w=20*np.random.rand(len(n))
plt.stem(w)
plt.show()
# Noisy Signal
s=s1+w
plt.stem(s)
plt.show()
# output Signal by FIR Filter
a=1;
yout = sig.lfilter(b,a,s)
plt.stem(yout)
plt.show()
```









Answer-5

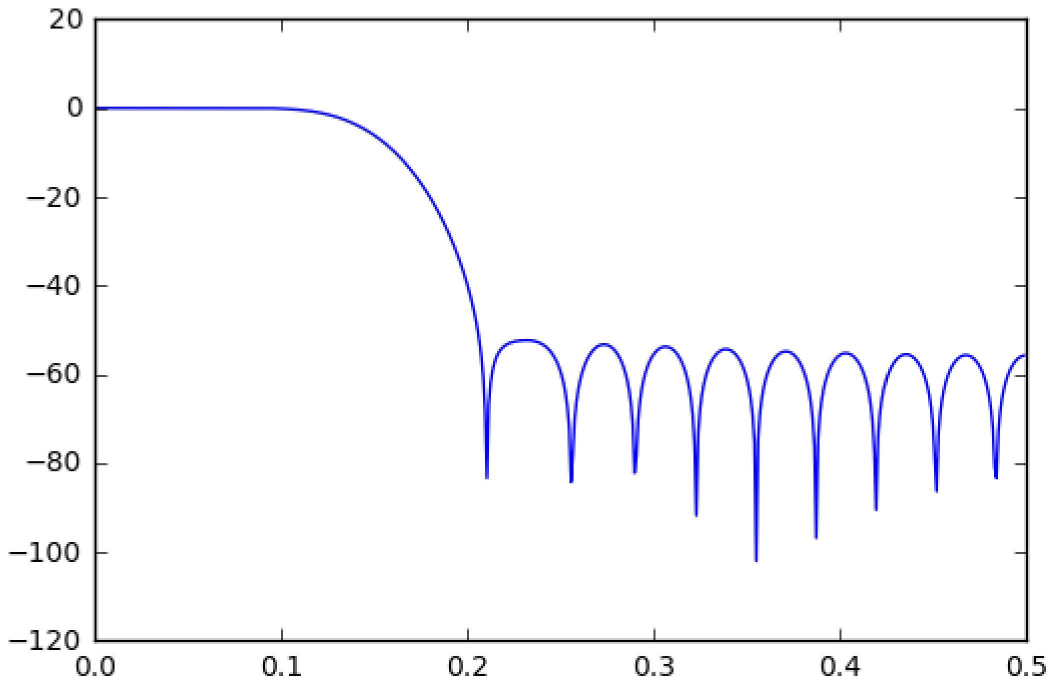
Q.5.(a)Determine and plot the filtered output signal by FIR Hamming window function for the following signal and also compare between noisy signal and filted output signal:
$x[n] = 0.1n + sin(0.1n\pi) + w[n]; 0 \leq n \leq 60$;Where $w[n]$ is zero mean uniformly distributed random noise

in $[-0.5\ 0.5]$.

(b)Repeat part(a) for Rectangular, hann, blackman, and bartlett windowing functions

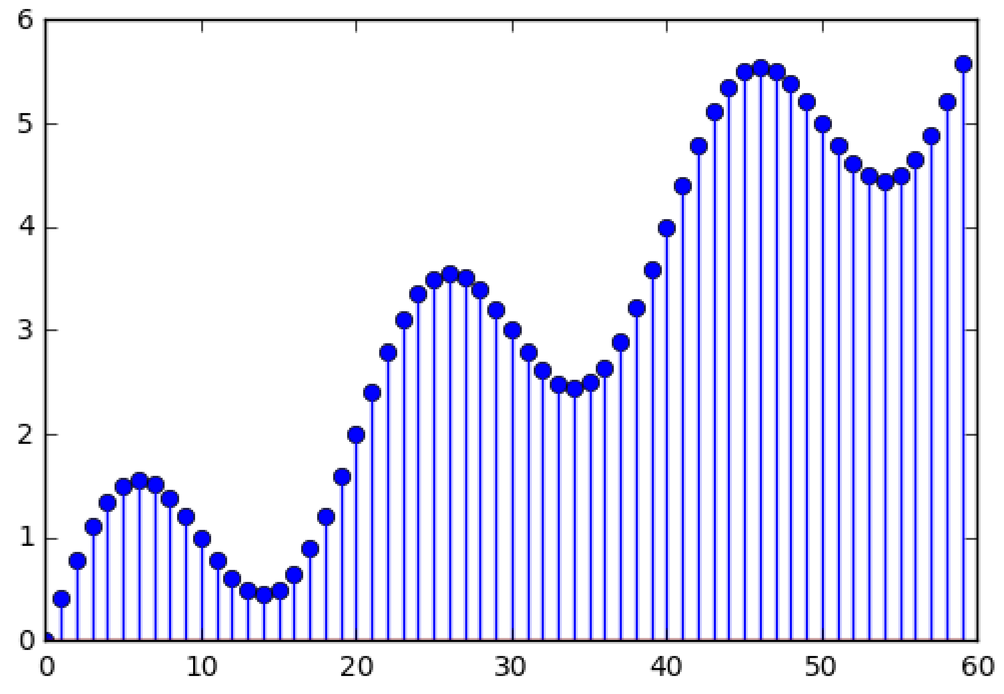
In [44]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

#Lowpass FIR filter
N = 31
b = sig.firwin(N, cutoff = 0.3, pass_zero=True, window = "hamming")
#Frequency response
w, h = sig.freqz(b)
plt.plot(w/(2*np.pi), 20*np.log10(abs(h)))
plt.show()
```



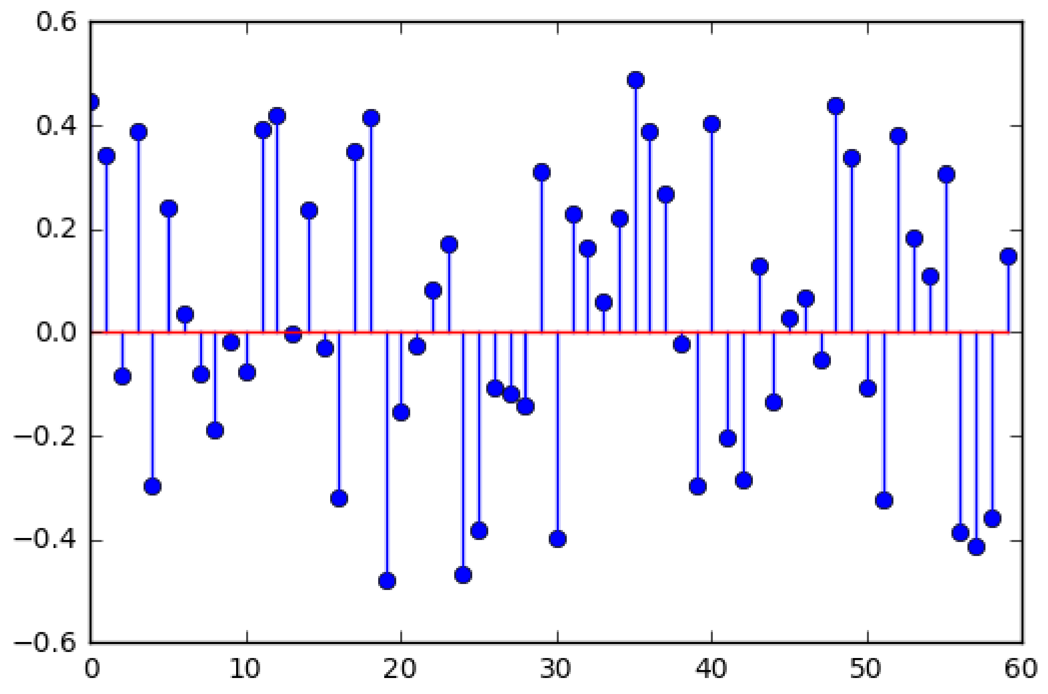
In [45]:

```
# Signal generation
n=range(0,60)
k=np.array(n)
x=(0.1*k) + np.sin(0.1*k*(np.pi))
plt.stem(n,x)
plt.show()
```



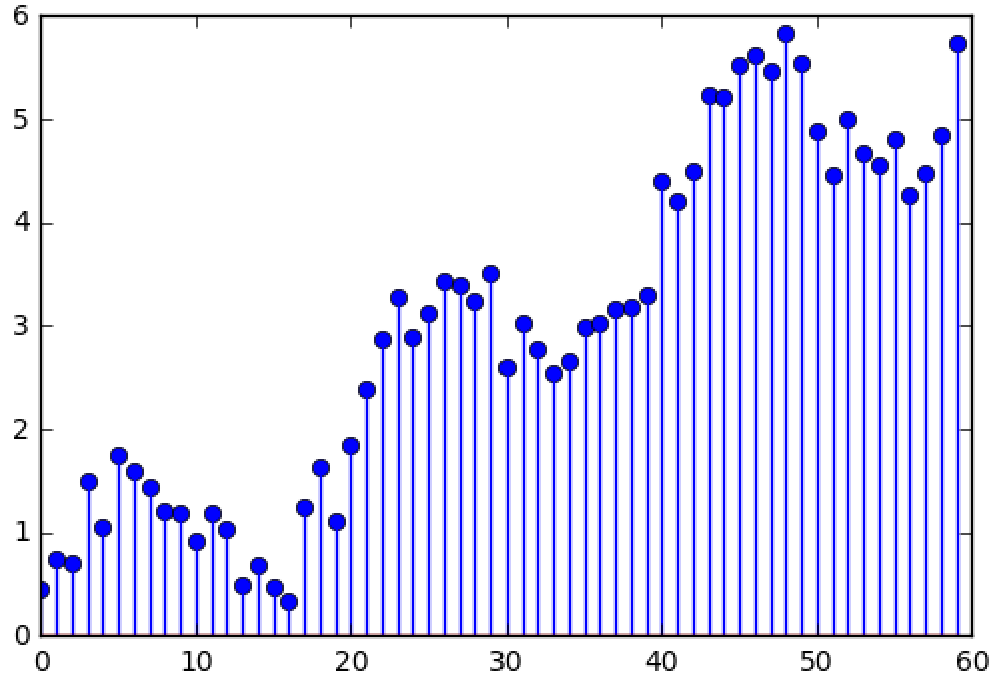
In [47]:

```
# uniform noise generation
w=np.random.uniform(-0.5, 0.5,60)
plt.stem(n,w)
plt.show()
```



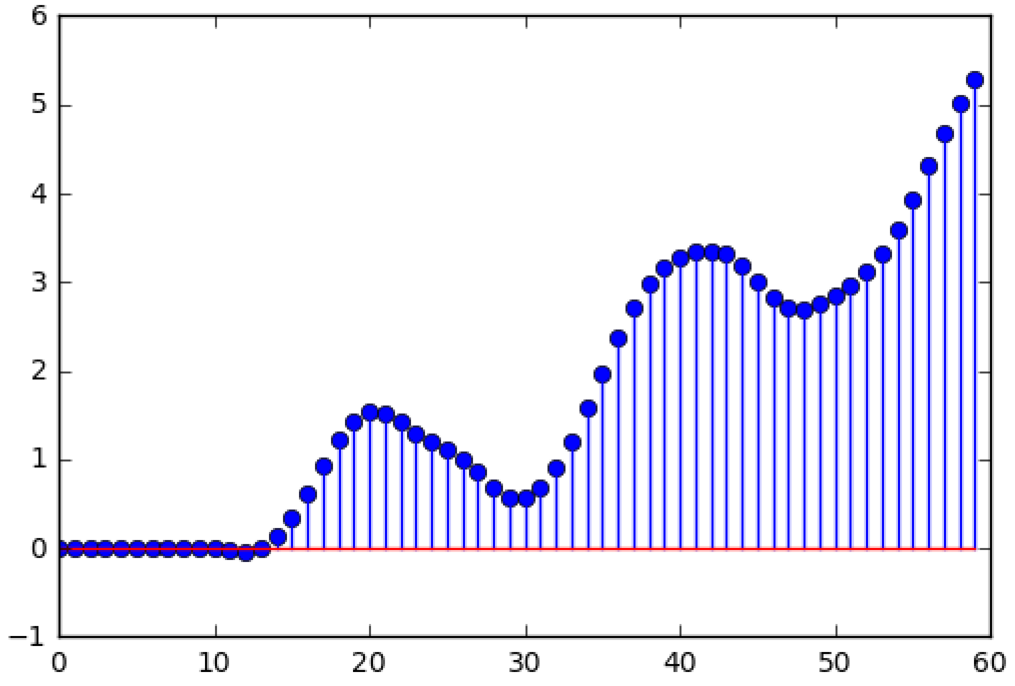
In [49]:

```
# Noisy signal
x1=(0.1*k) + np.sin(0.1*k*(np.pi))+ w
plt.stem(n,x1)
plt.show()
```



In [52]:

```
# Filtered output signal
a=1
yout = sig.lfilter(b,a,x1)
plt.stem(yout)
plt.show()
```

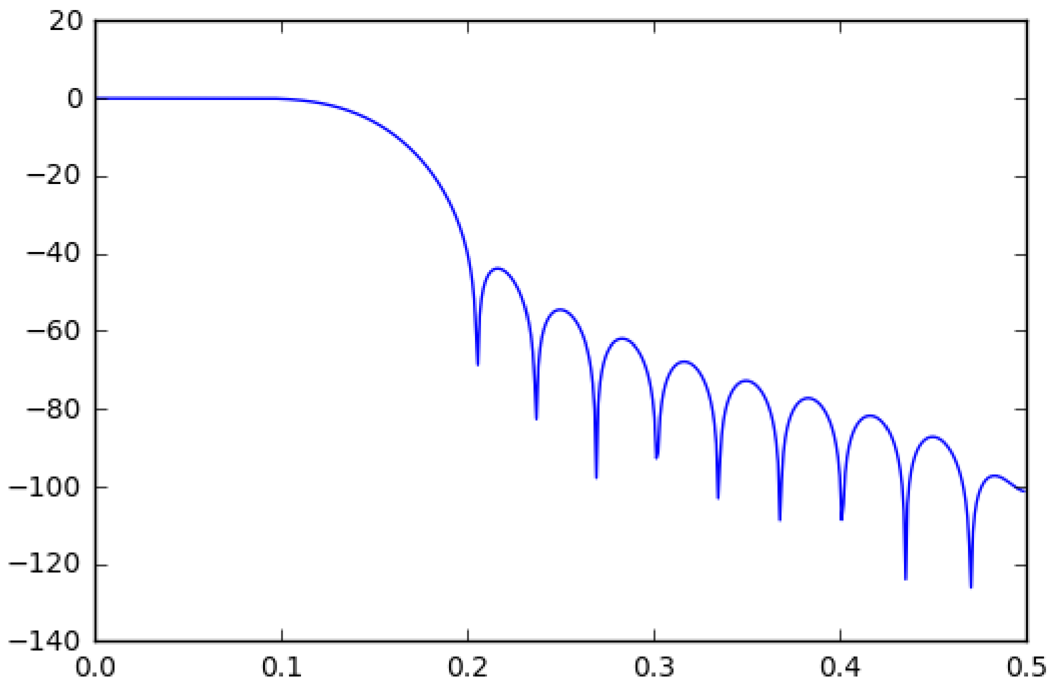


Answer-6

Q.6. Record your own voice for half a minute and observe the time domain waveform (It can be done using a Audacity sound recording tool in the computer. Please save in *.wav format, upload it and read it using wavread command in Python for further processing). Please look https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html) for documentation
(a) Find the sampling rate used by the recorder
(b) Generate noisy signal of your recoded signal by adding zero mean uniformly distributed random noise in $[-0.5\ 0.5]$.
(c) Determine and plot the filtered output signal by FIR Filter using Hanning window and also compare between noisy signal and filted output signal

In [53]:

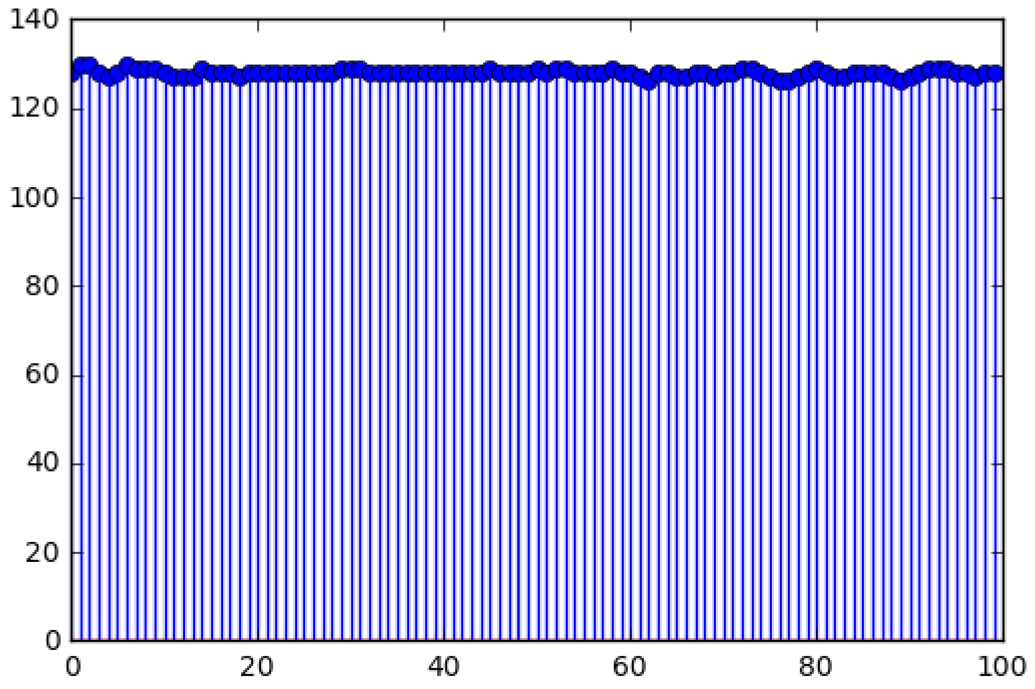
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
import scipy.io.wavfile as wav

#Lowpass FIR filter
N = 31
b = sig.firwin(N, cutoff = 0.3, pass_zero=True, window = "hann")
#Frequency response
w, h = sig.freqz(b)
plt.plot(w/(2*np.pi), 20*np.log10(abs(h)))
plt.show()
```



In [54]:

```
# Signal generation
r,d=wav.read('Funny.wav') # Return the sample rate {r} (in samples/sec) and data(d) from a
print(r)
x=d[100:200]
print(len(x))
plt.stem(x)
plt.show()
n=range(0,len(x))
```

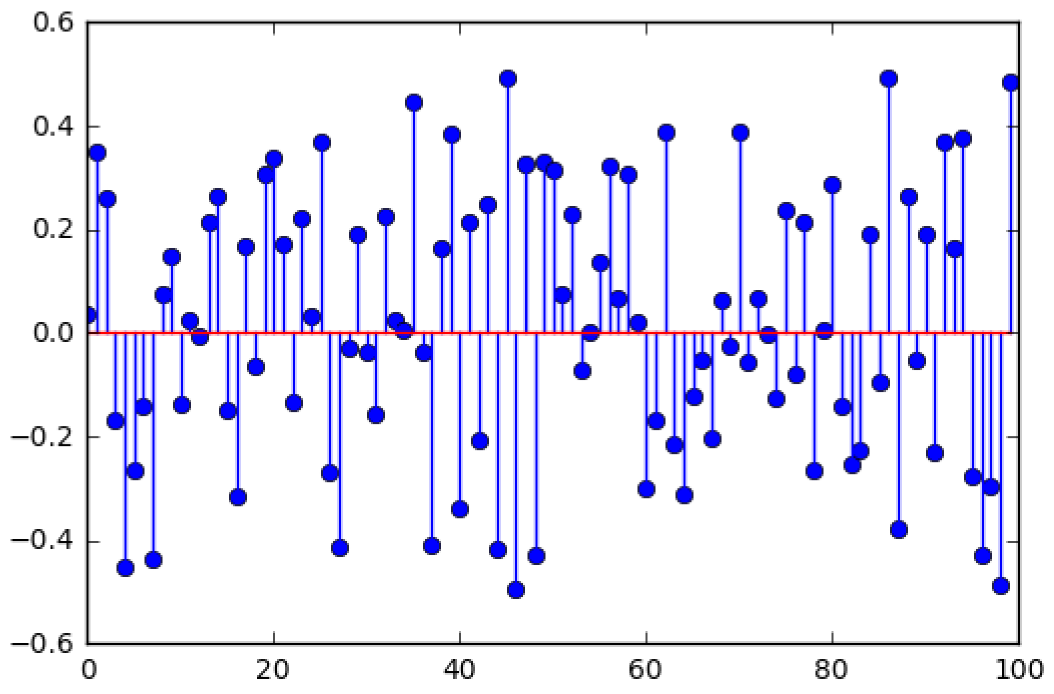
```
22050
100
```



In [56]:

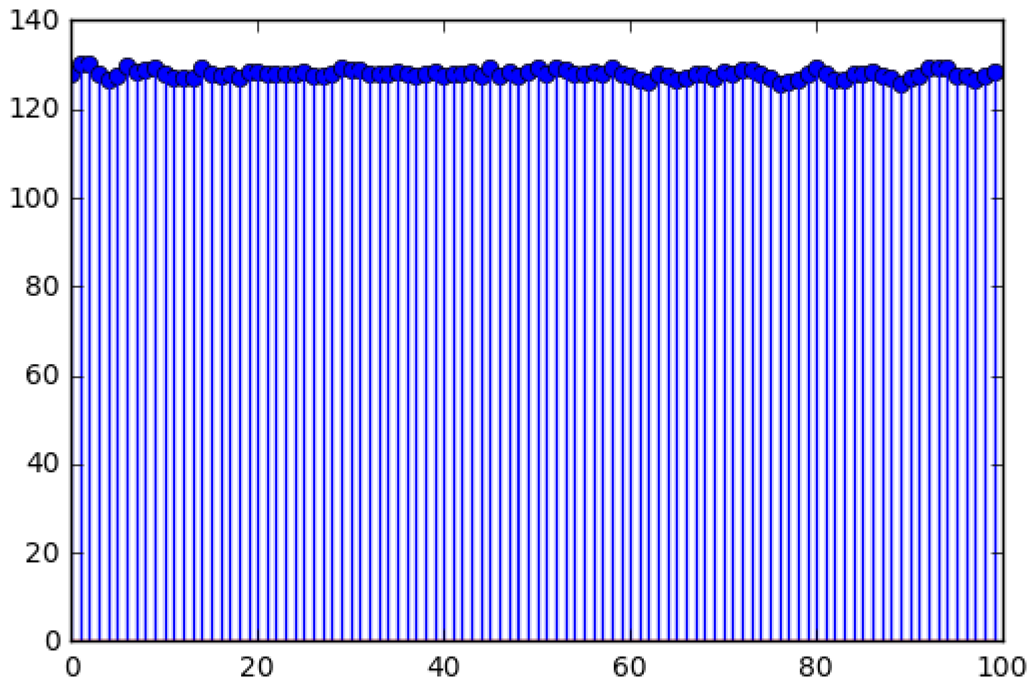
```
# uniform noise generation
w=np.random.uniform(-0.5, 0.5,len(x))
print(len(w))
plt.stem(w)
plt.show()
```

```
100
```



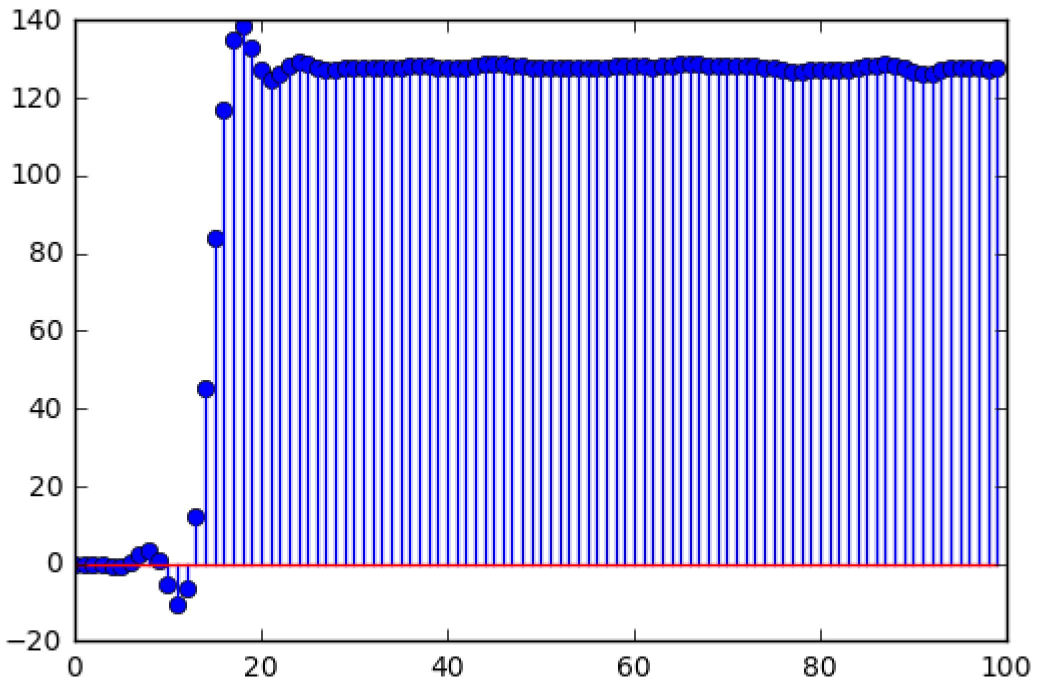
In [58]:

```
# Noisy signal
x1= x + w
plt.stem(x1)
plt.show()
```



In [60]:

```
# Filtered output signal
a=1
yout = sig.lfilter(b,a,x1)
plt.stem(yout)
plt.show()
```



Answer 7

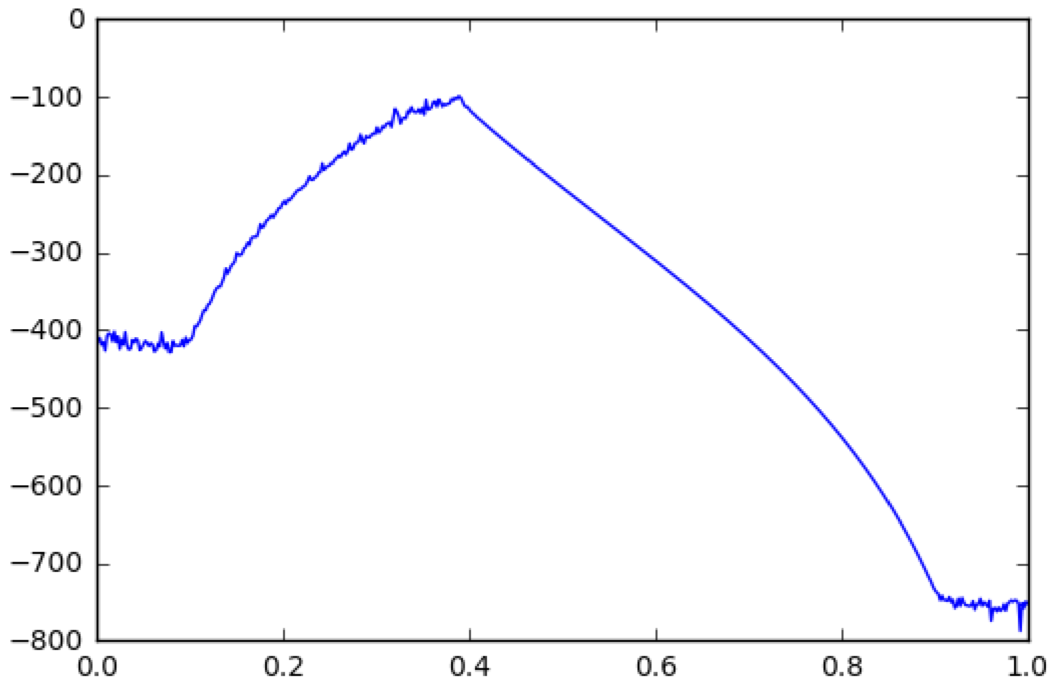
Q.7. Determine and plot frequecy response of Low Butterworth and Chebyshev-I IIR Filters. The details about IIR Filter Design Python function{scipy.signal.iirdesign(wp, ws, gpass, gstop, analog=False, ftype='ellip', output='ba', fs=None)} are given: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.iirdesign.html#scipy.signal.iirdesign (https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.iirdesign.html#scipy.signal.iirdesign) Passband and stopband edge frequencies. For digital filters, these are in the same units as fs. By default, fs is

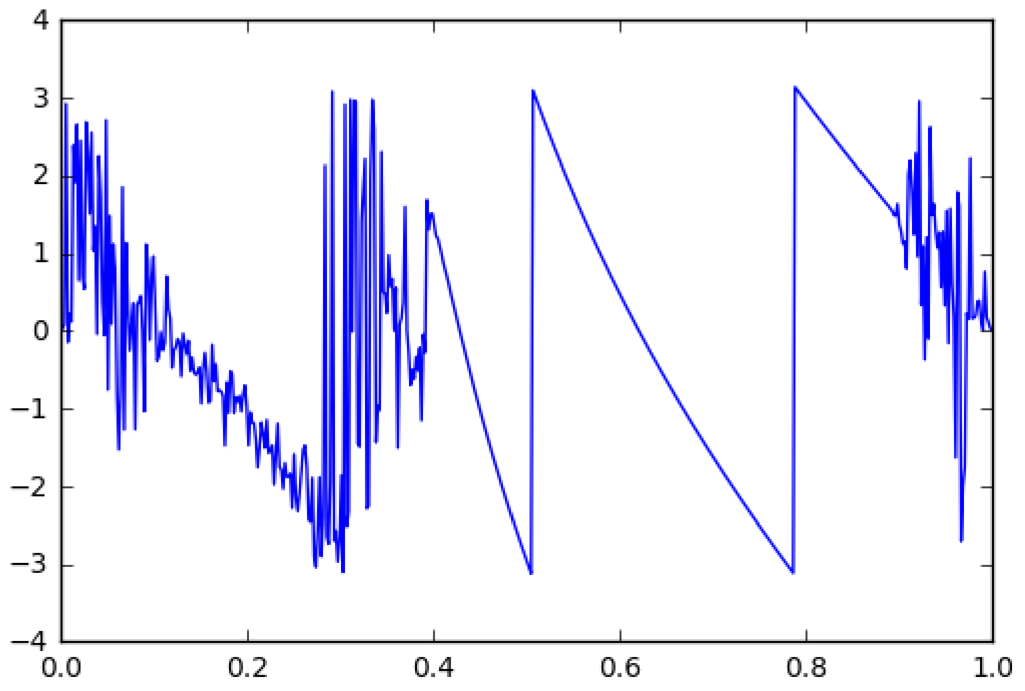
2 half-cycles/sample, so these are normalized from 0 to 1, where 1 is the Nyquist frequency. For example:#Lowpass: wp = 0.2, ws = 0.3;~#Highpass: wp = 0.3, ws = 0.2;~#Bandpass: wp = [0.2, 0.5], ws = [0.1, 0.6];#Bandstop: wp = [0.1, 0.6], ws = [0.2, 0.5]

#The type of IIR filter to design:#Butterworth: ‘butter’;#Chebyshev I : ‘cheby1’;#Chebyshev II : ‘cheby2’; #Elliptic: ‘ellip’; Bessel/Thomson: ‘bessel’

In [66]:

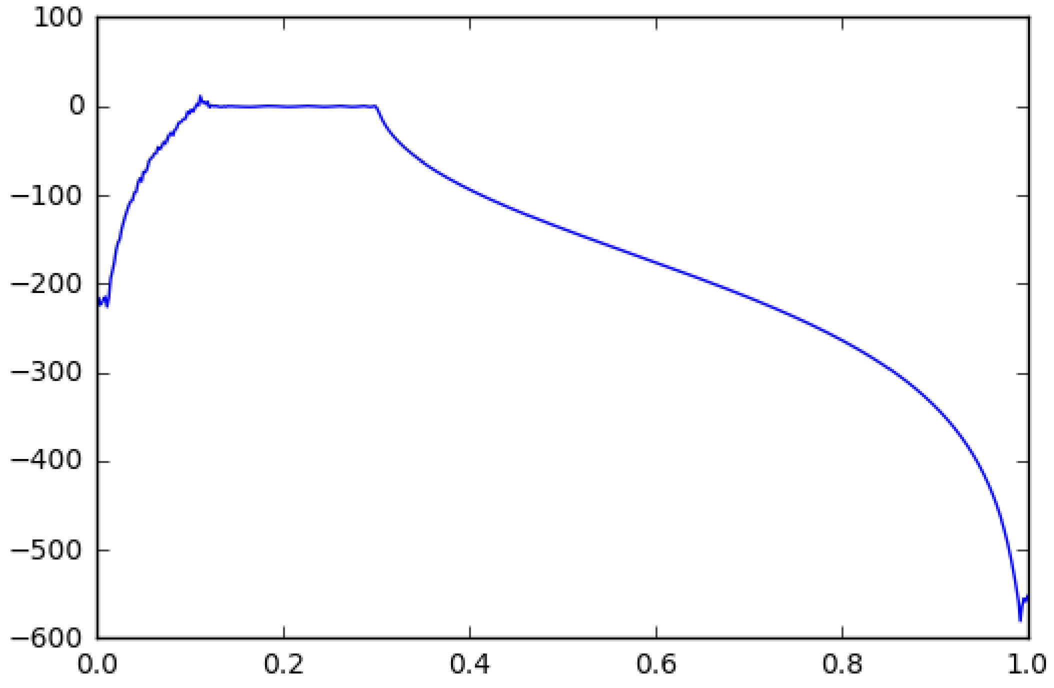
```
# Low Pass butterworth IIR Filter
b,a = sig.iirdesign(wp = [0.05, 0.3], ws= [0.02, 0.35], gstop= 60, gpass=1, ftype='butter')
w,h = sig.freqz(b,a)
h_dB = 20 *np.log10 (abs(h))
plt.plot(w/max(w),h_dB)
plt.show()
angles = np.angle(h)
plt.plot(w/max(w),angles)
plt.show()
```

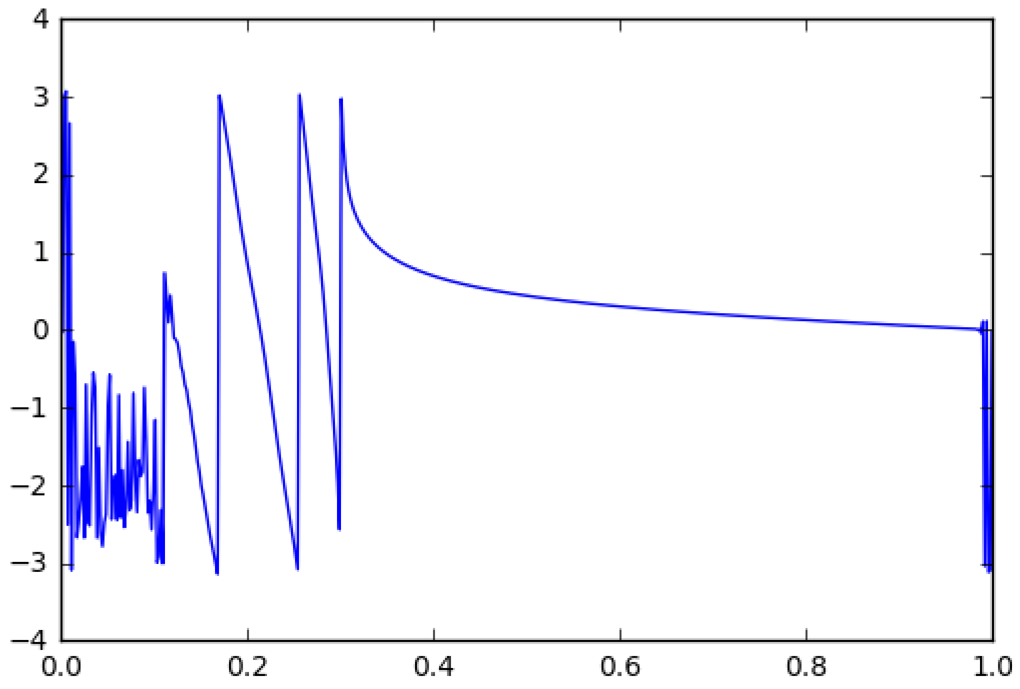




In [67]:

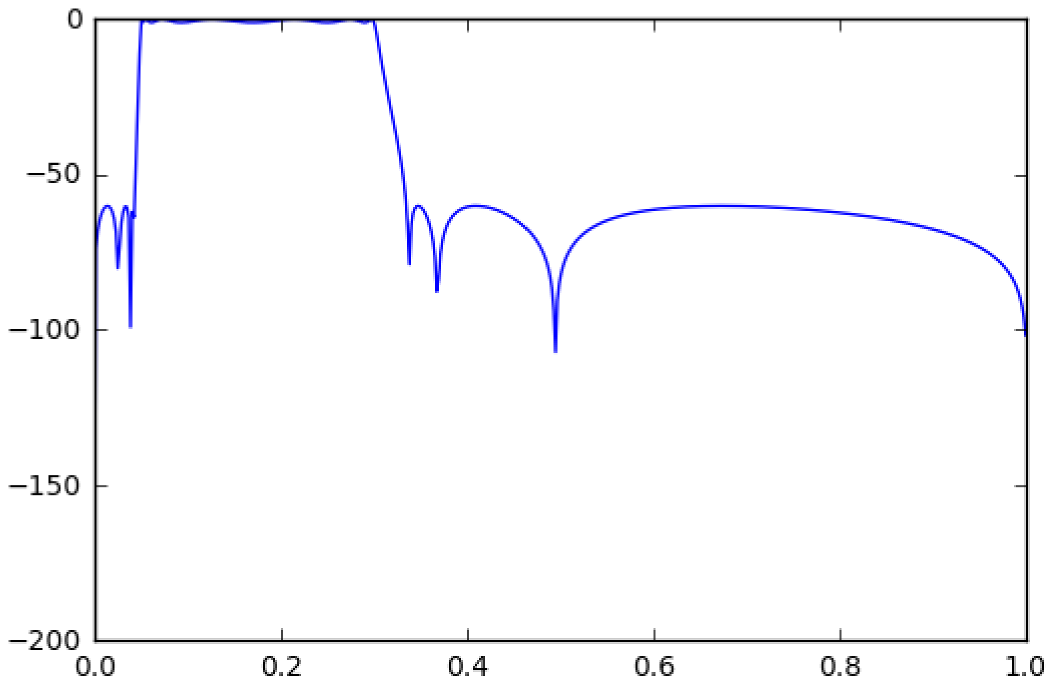
```
# Low Pass Chebyshev-I IIR Filter
b,a = sig.iirdesign(wp = [0.05, 0.3], ws= [0.02, 0.35], gstop= 60, gpass=1, ftype='cheby1')
w,h = sig.freqz(b,a)
h_dB = 20 *np.log10 (abs(h))
plt.plot(w/max(w),h_dB)
plt.show()
angles = np.angle(h)
plt.plot(w/max(w),angles)
plt.show()
```

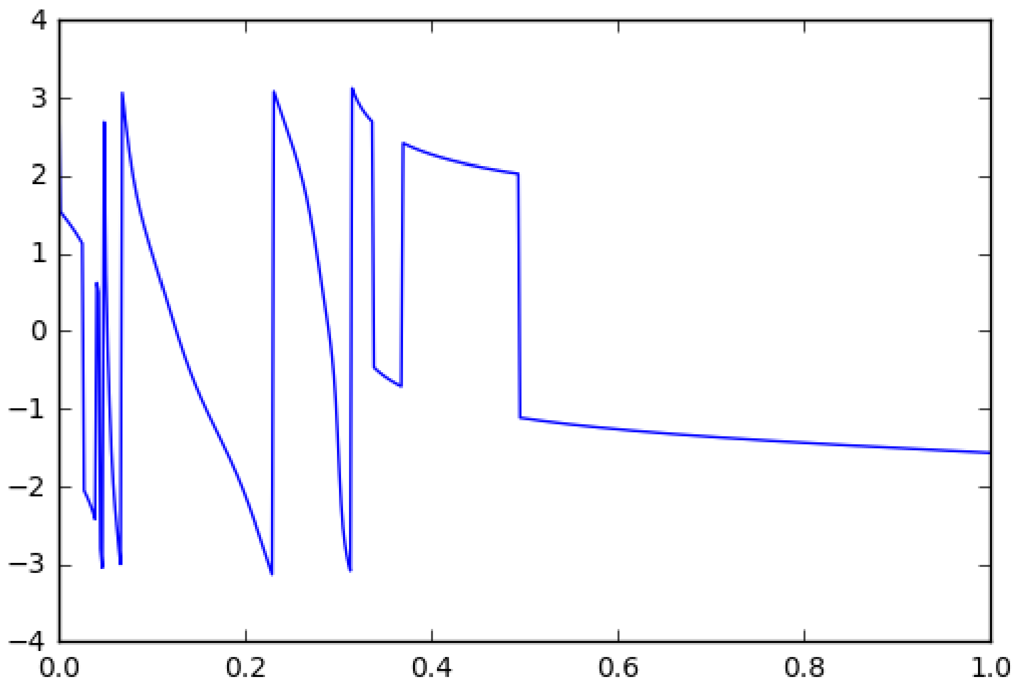




In [68]:

```
# Low pass Elliptic IIR  filter
b,a = sig.iirdesign(wp = [0.05, 0.3], ws= [0.02, 0.35], gstop= 60, gpass=1, ftype='ellip')
w,h = sig.freqz(b,a)
h_dB = 20 *np.log10 (abs(h))
plt.plot(w/max(w),h_dB)
plt.show()
angles = np.angle(h)
plt.plot(w/max(w),angles)
plt.show()
```





Answer 8

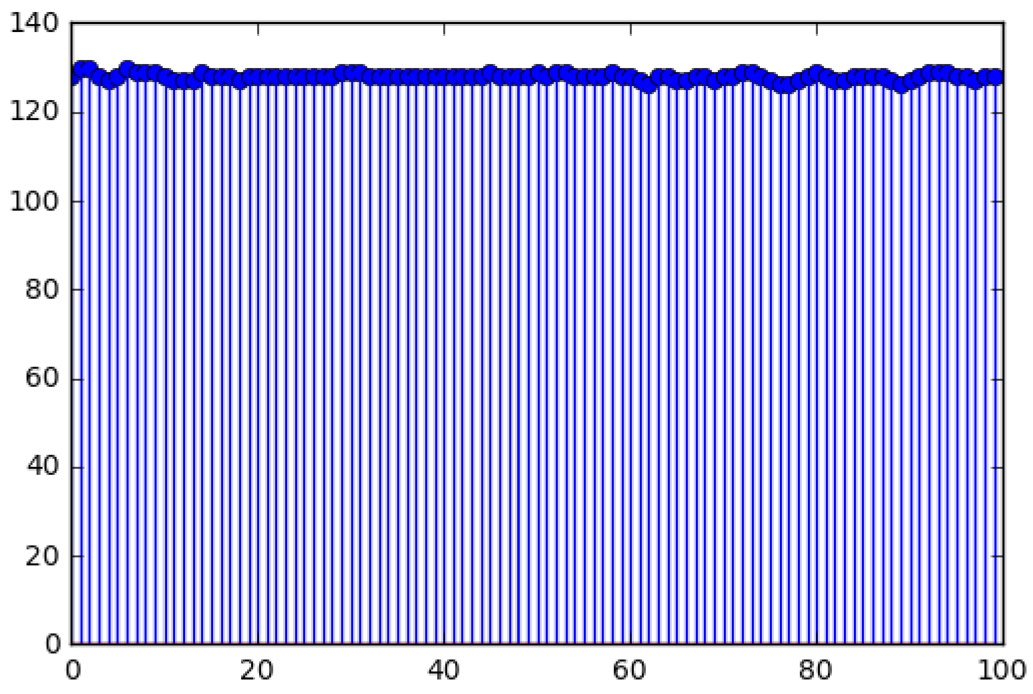
Q.8. Record your own voice for half a minute and observe the time domain waveform (It can be done using a

Audacity sound recording tool in the computer. Please save in *.wav format, upload it and read it using wavread command in Python for further processing). Please look https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html) for documentation
(a) Find the sampling rate used by the recorder
(b) Generate noisy signal of your recoded signal by adding zero mean uniformly distributed random noise in $[-0.5\ 0.5]$.
(c) Determine and plot the filtered output signal by IIR Filter and also compare between noisy signal and filted output signal

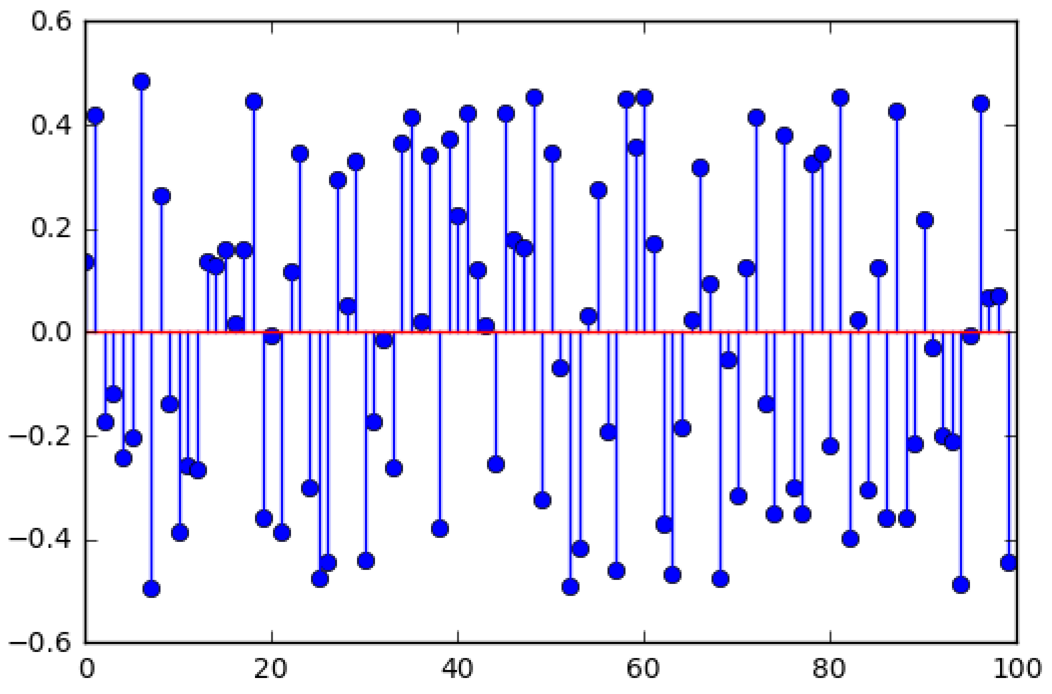
In [69]:

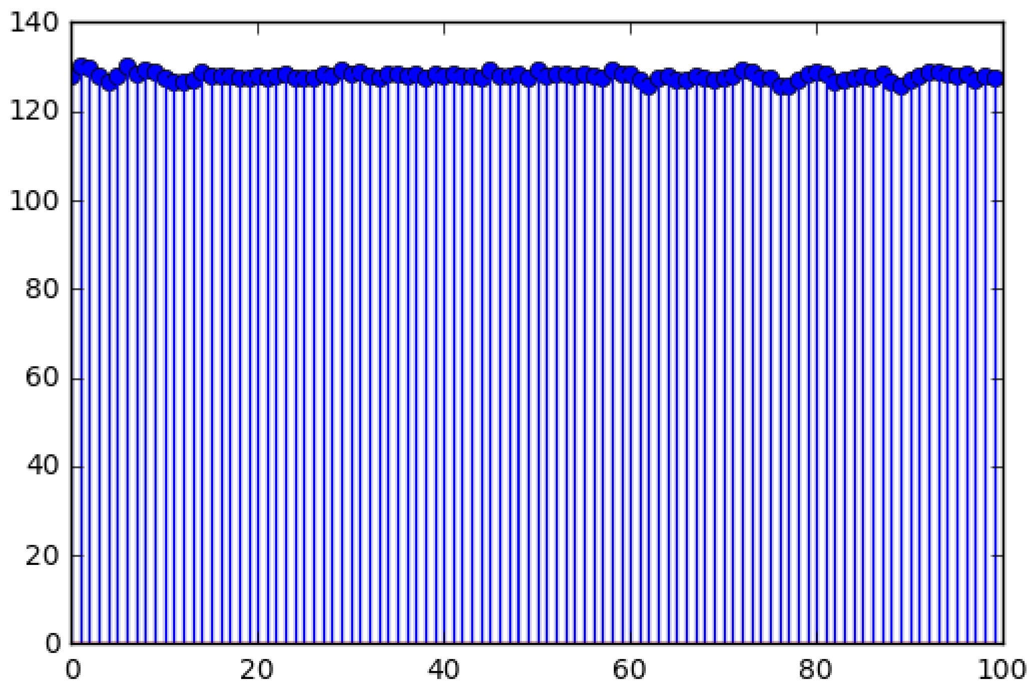
```
import numpy as np
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt
import scipy.signal as sig

# Signal generation
r,d=wav.read('Funny.wav') # Return the sample rate {r} (in samples/sec) and data(d) from a
print(r)
x=d[100:200]
print(len(x))
plt.stem(x)
plt.show()
n=range(0,len(x))
# Noise Signal
# uniform noise generation
w=np.random.uniform(-0.5, 0.5,len(x))
print(len(w))
plt.stem(w)
plt.show()
# Noisy signal
x1= x + w
plt.stem(x1)
plt.show()
```

```
22050
100
```



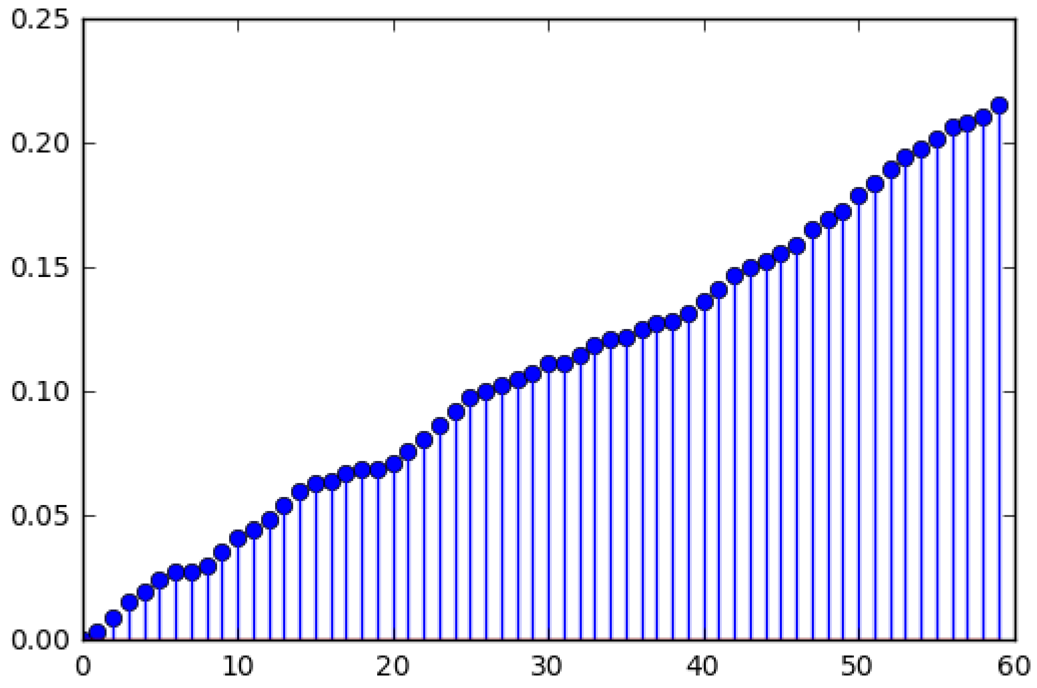
```
100
```


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


140
120
100
80
60
40
20
0
0
20
40
60
80
100

In [72]:

```
# Create a 1st order butterworth filter
Fs = 288e6
Fpwm = 32e3
b,a = sig.butter(1, (Fpwm/2)/(Fs/2))
# filter the output of the PWM, yn
#yout = signal.filtfilt(b,a,s)
yout = sig.lfilter(b,a,s)
plt.stem(yout)
plt.show()
```



In [ ]: