# Lab 1: Introduction to Python, Digital Frequency and Spectrum

import numpy as np

import scipy.fftpack

import matplotlib.pyplot as plt

plt.close("all")

A=10

fa=1000 #Analogue Freq

fs=100000.0 #Sampling Freq

N=200 #No. of display samples

n = np.arange(0, N-1) #Discrete time values

t=1000\*n/fs

Q=2\*np.pi\*fa/fs # Normalised frequency

tone=A\*np.sin(n\*Q)

plt.figure(1)

plt.plot(t,tone)

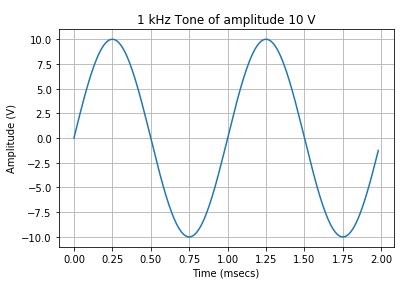
plt.title('1 kHz Tone of amplitude 10 V')

plt.xlabel('Time (msecs)')

plt.ylabel('Amplitude (V)')

plt.grid()

plt.show()



sig1=5\*np.cos(n\*Q)+np.sin(n\*10\*Q)

plt.figure(2)

plt.plot(t,sig1)

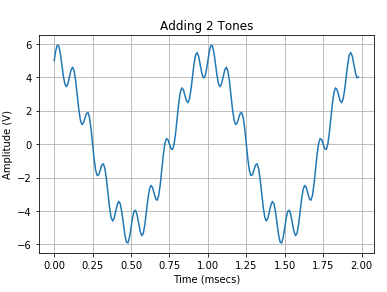
plt.title('Adding 2 Tones')

plt.xlabel('Time (msecs)')

plt.ylabel('Amplitude (V)')

plt.grid()

plt.show()



sig2=1\*np.cos(n\*Q)\*np.cos(n\*10\*Q)

plt.figure(3)

plt.plot(t,sig2)

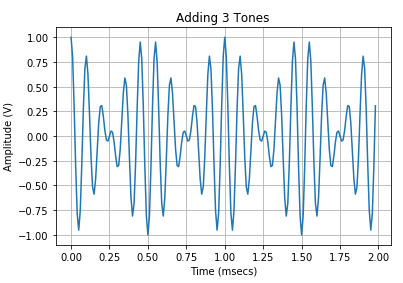
plt.title('Adding 3 Tones')

plt.xlabel('Time (msecs)')

plt.ylabel('Amplitude (V)')

plt.grid()

plt.show()



#-------------------Generating the FFT of the signal---------------------------

NFFT=1024 # No. of values in FFT

M = 2\*np.abs(scipy.fftpack.fft(sig1,NFFT))/N

M = M[0:int(NFFT/2)] #slicing operation to avoid mirroring

freq = np.arange(0,NFFT/2) #frequency vector

freq = freq\*fs/NFFT

plt.figure(4)

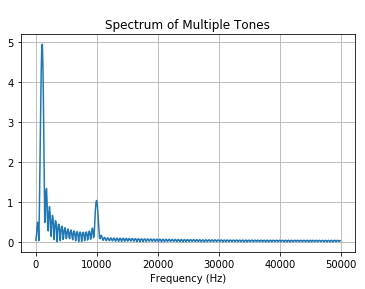
plt.plot(freq,M)

plt.title('Spectrum of Multiple Tones')

plt.xlabel('Frequency (Hz)')

plt.grid()

plt.show()



plt.figure(7)

plt.plot(scipy.hamming(N-1))

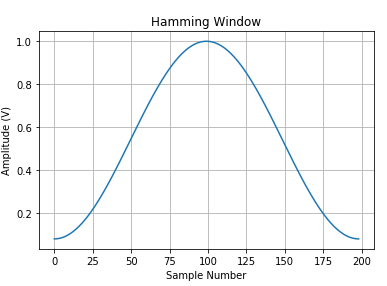
plt.title('Hamming Window')

plt.xlabel('Sample Number')

plt.ylabel('Amplitude (V)')

plt.grid()

plt.show()



Multiplying FFT by the hamming signal to reduce the spectrum leakage.

#-------------------Multiplying FFT by the hamming Signal---------------------------

NFFT=1024 # No. of values in FFT

M = 2\*np.abs(scipy.fftpack.fft(wsig1,NFFT))/N

M = M[0:int(NFFT/2)] #slicing operation to avoid mirroring

freq = np.arange(0,NFFT/2) #frequency vector

freq = freq\*fs/NFFT

plt.figure(4)

plt.plot(freq,M)

plt.title('Spectrum of Multiple Tones')

plt.xlabel('Frequency (Hz)')

plt.grid()

plt.show()

