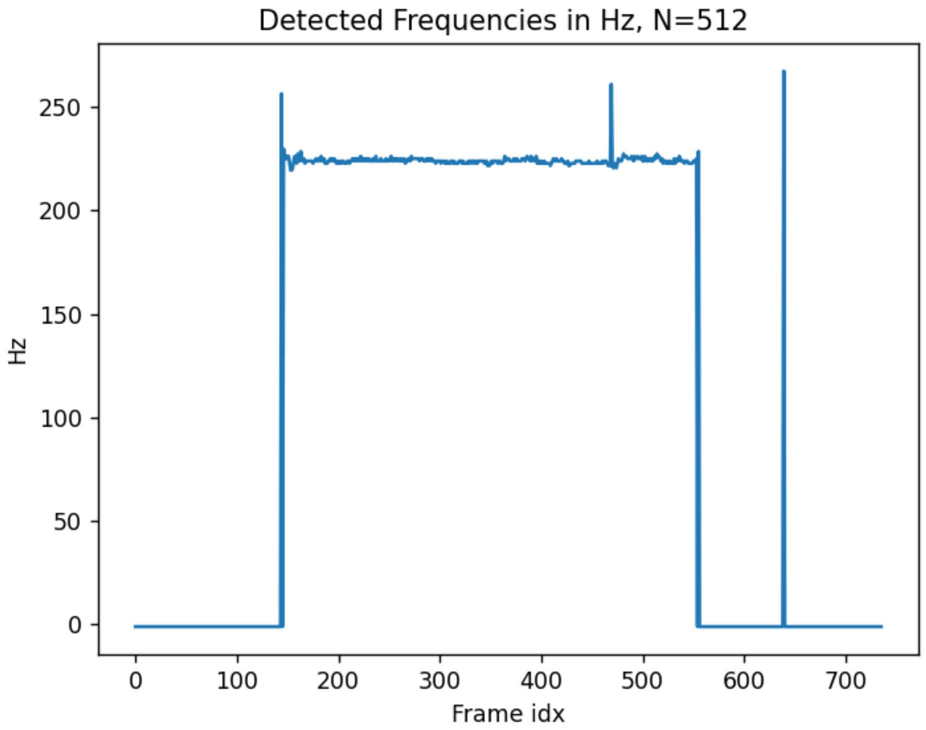
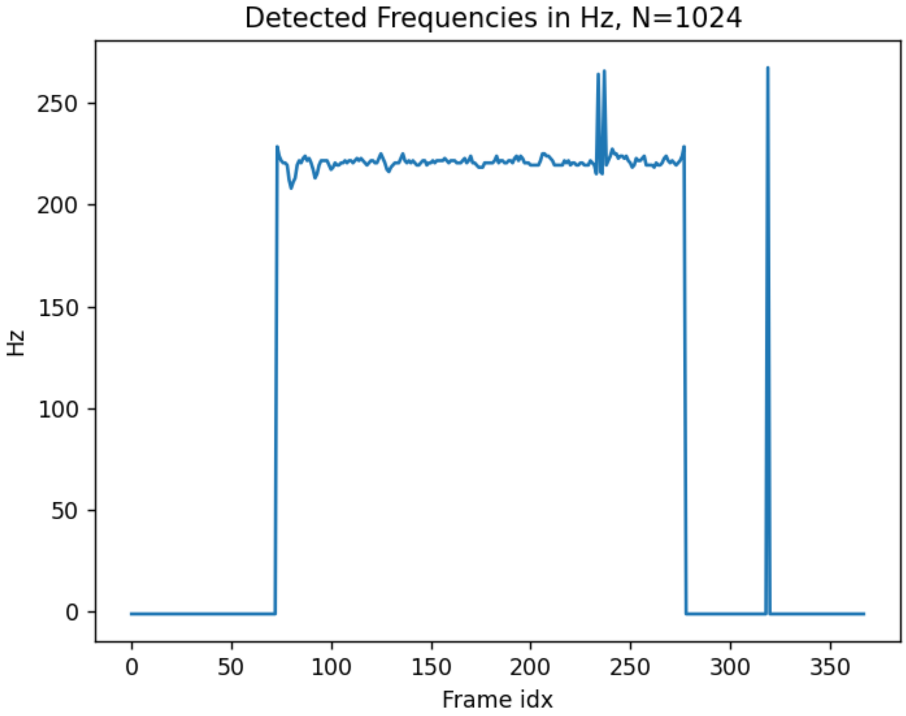
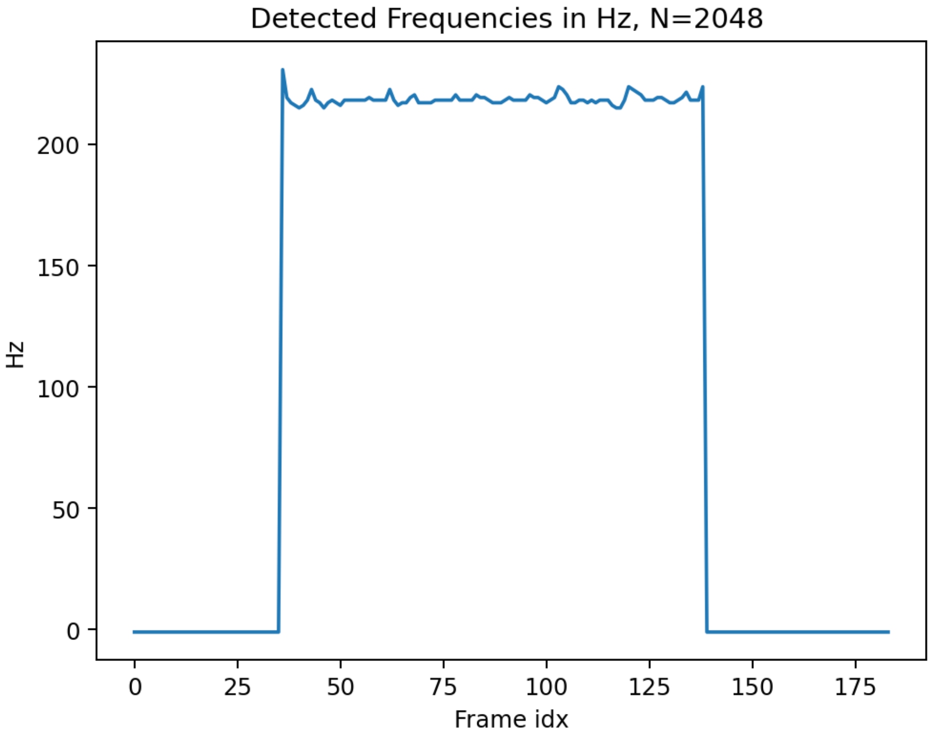
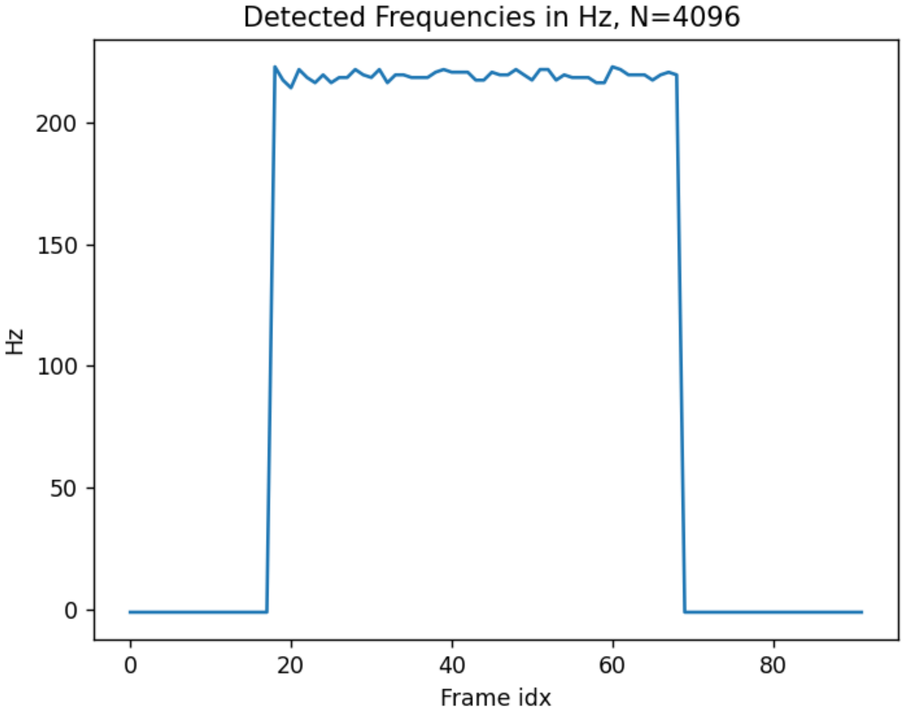
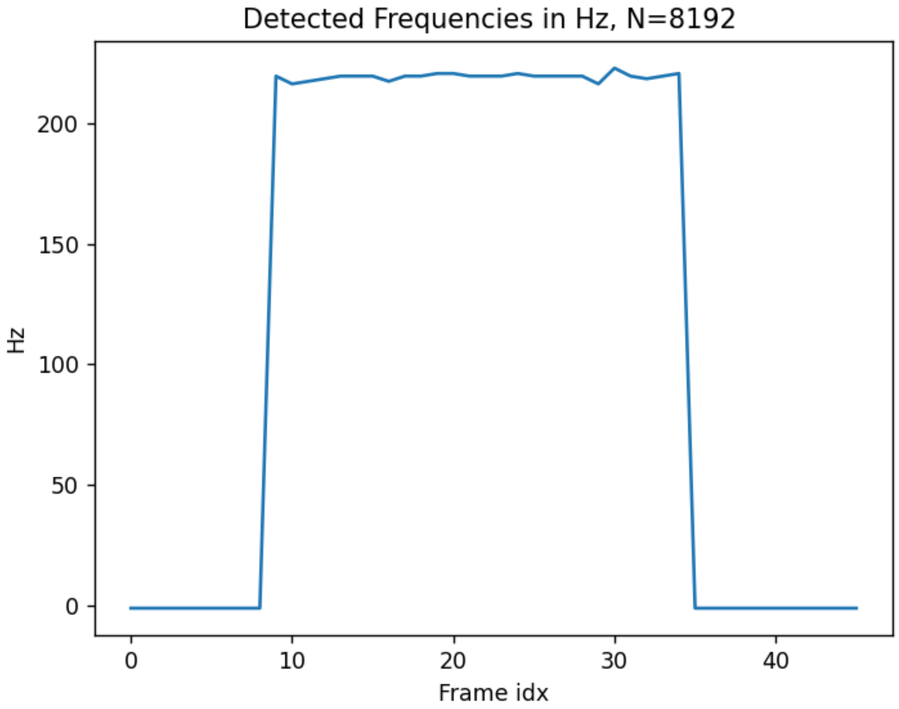

Detected Frequencies in Hz, N=512

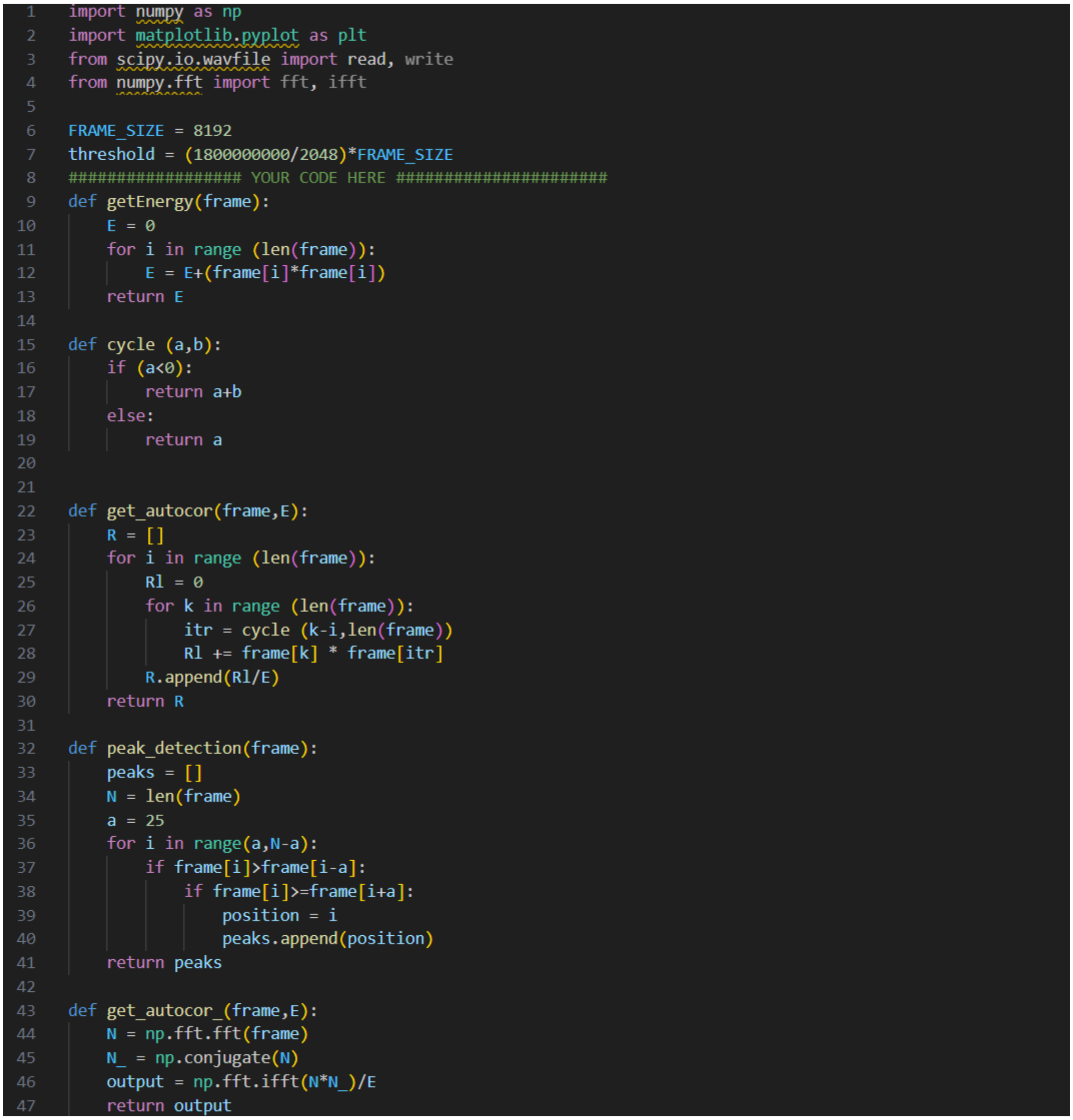

Detected Frequencies in Hz, N=1024

.

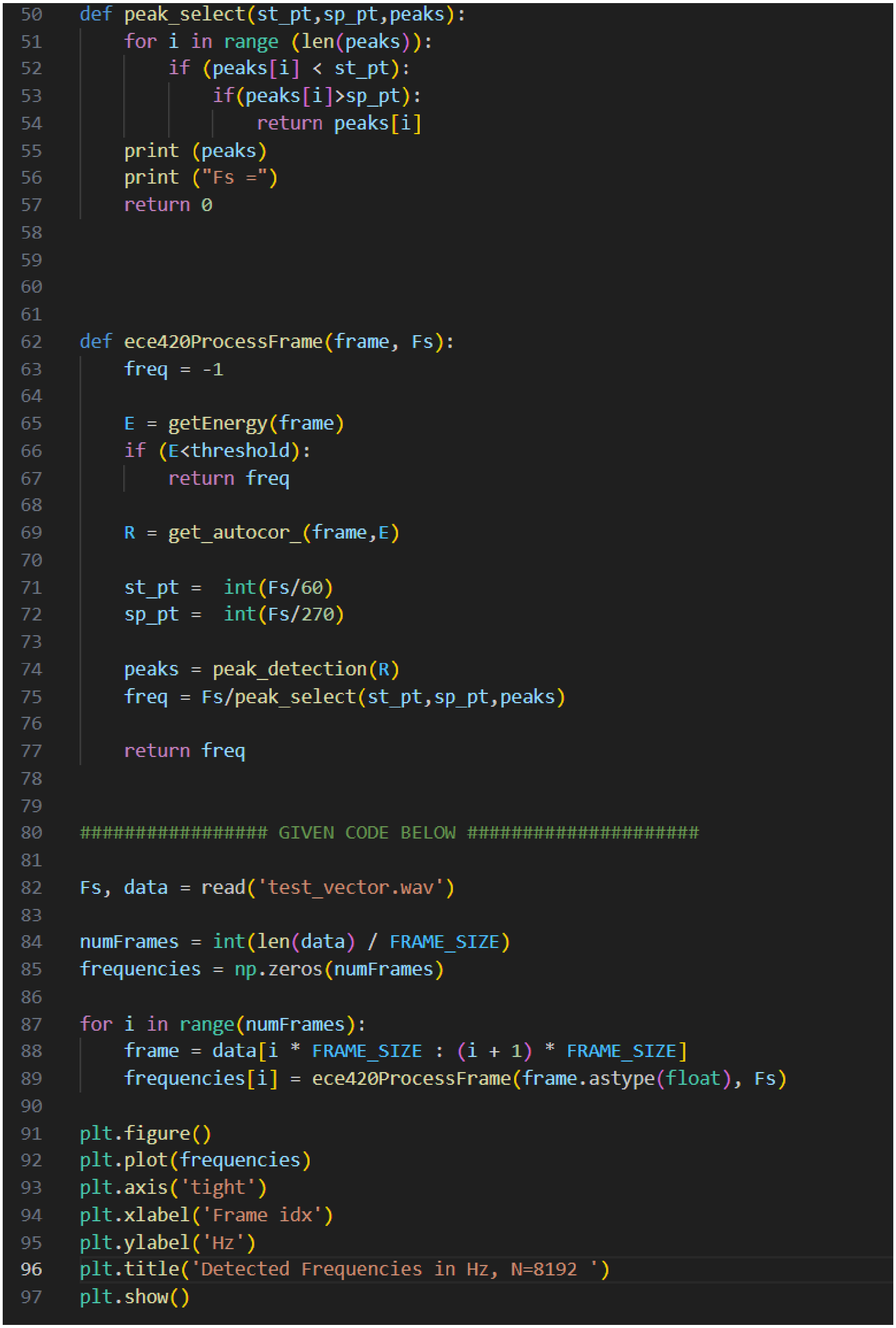

Detected Frequencies in Hz, N=2048
Hz
0
50
100
150
200
Frame idx
0
25
50
75
100
125
150
175


Detected Frequencies in Hz, N=4096
Hz
0
50
100
150
200
Frame idx
0
20
40
60
80


Detected Frequencies in Hz, N=8192
Hz
0
50
100
150
200
Frame idx
0
10
20
30
40



```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import read, write
from numpy.fft import fft, ifft

FRAME_SIZE = 8192
threshold = (1800000000/2048)*FRAME_SIZE
################## YOUR CODE HERE ######################
def getEnergy(frame):
    E = 0
    for i in range (len(frame)):
        E = E+(frame[i]*frame[i])
    return E

def cycle (a,b):
    if (a<0):
        return a+b
    else:
        return a


def get_autocor(frame,E):
    R = []
    for i in range (len(frame)):
        Rl = 0
        for k in range (len(frame)):
            itr = cycle (k-i,len(frame))
            Rl += frame[k] * frame[itr]
        R.append(Rl/E)
    return R

def peak_detection(frame):
    peaks = []
    N = len(frame)
    a = 25
    for i in range(a,N-a):
        if frame[i]>frame[i-a]:
            if frame[i]>=frame[i+a]:
                position = i
                peaks.append(position)
    return peaks

def get_autocor_(frame,E):
    N = np.fft.fft(frame)
    N_ = np.conjugate(N)
    output = np.fft.ifft(N*N_)/E
    return output
```



```python
def peak_select(st_pt,sp_pt,peaks):
    for i in range (len(peaks)):
        if (peaks[i] < st_pt):
            if(peaks[i]>sp_pt):
                return peaks[i]
    print (peaks)
    print ("Fs =")
    return 0




def ece420ProcessFrame(frame, Fs):
    freq = -1

    E = getEnergy(frame)
    if (E<threshold):
        return freq

    R = get_autocor_(frame,E)

    st_pt =  int(Fs/60)
    sp_pt =  int(Fs/270)

    peaks = peak_detection(R)
    freq = Fs/peak_select(st_pt,sp_pt,peaks)

    return freq


################# GIVEN CODE BELOW #####################

Fs, data = read('test_vector.wav')

numFrames = int(len(data) / FRAME_SIZE)
frequencies = np.zeros(numFrames)

for i in range(numFrames):
    frame = data[i * FRAME_SIZE : (i + 1) * FRAME_SIZE]
    frequencies[i] = ece420ProcessFrame(frame.astype(float), Fs)

plt.figure()
plt.plot(frequencies)
plt.axis('tight')
plt.xlabel('Frame idx')
plt.ylabel('Hz')
plt.title('Detected Frequencies in Hz, N=8192 ')
plt.show()
```

.

Question 1. I would ignore pitches outside of normal human voice range as shown in code (60Hz – 270Hz). Then I would take the first peak in that range.

Question 2. Same as above, I would ignore frequencies outside of human voice range, given the first peak and its surrounding is close to sampling frequency which is in the kHz range, we can ignore them.

Question 3. Since the lower bound of human voice is around 85Hz, this would give us around 11ms period. 40ms would insure we capture at least 1 period of the fundamental frequency and provide enough time resolution.