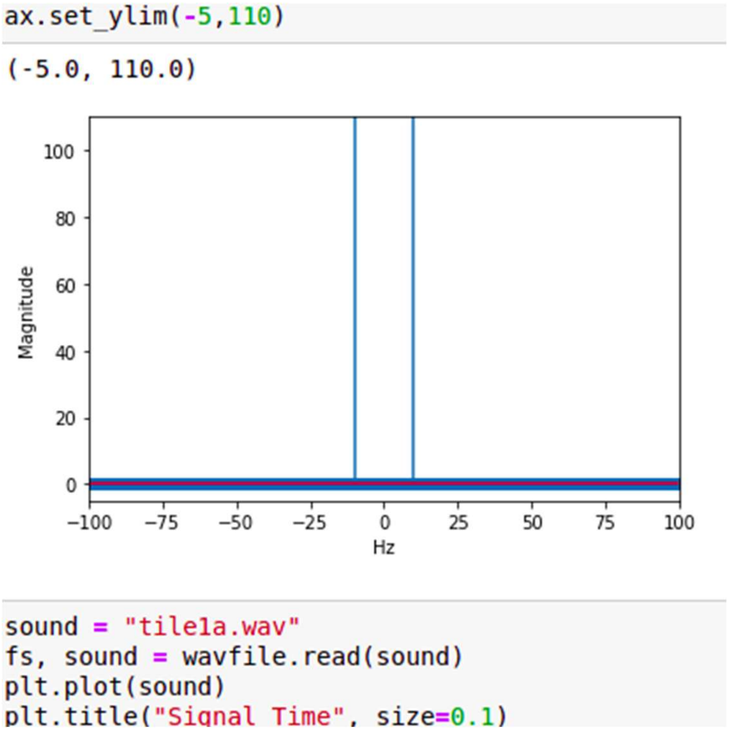
1a)

```
ex = fftpack.fft(x)
frequency = fftpack.fftfreq(len(x)) * N
fig,ax=plt.subplots()
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

1b) Did not vary much. However it is nearby -10 Hz to 10Hz



```
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

```
sound = "tile1a.wav"
fs, sound = wavfile.read(sound)
plt.plot(sound)
plt.title("Signal Time", size=0.1)
```

1c)

```
import matplotlib.pyplot as plt
from scipy import fftpack
from scipy.io import wavfile
from scipy.fftpack import fft
import numpy as np
samplerate, data = wavfile.read('tile1a.wav')
fft_out = fft(data)
%matplotlib inline
plt.plot(data,np.abs(fft_out))
plt.show()
```

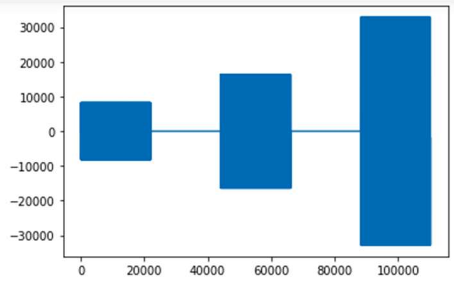
```
a= 4000
N = 4000
t = np.linspace(0,2,2 * N, endpoint=False)
x = np.sin(f*2*np.pi*t)
fig,ax=plt.subplots()
ax.plot(t,x)
ax.set_xlabel('Hz')
ax.set_ylabel('Sig')

ex = fftpack.fft(x)
frequency = fftpack.fftfreq(len(x)) * N
fig,ax=plt.subplots()
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)

samplerate, data = wavfile.read('cos_1khz_pulse_20msec.wav')
fft_out = fft(data)
%matplotlib inline
plt.plot(data,np.abs(fft_out))
plt.show()

a= 100
N = 200
t = np.linspace(0,2,2 * N, endpoint=False)
x = np.sin(f*2*np.pi*t)
fig,ax=plt.subplots()
ax.plot(t,x)
ax.set_xlabel('Hz')
ax.set_ylabel('Sig')

ex = fftpack.fft(x)
frequency = fftpack.fftfreq(len(x)) * N
fig,ax=plt.subplots()
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

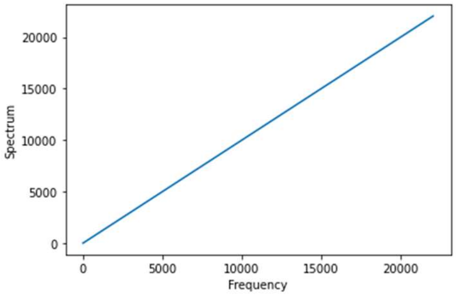
2a&2b)

```
import math
import wave
import struct
```

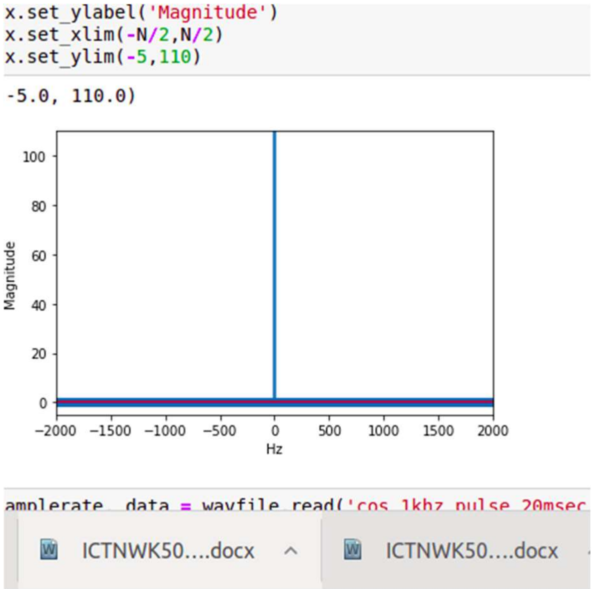
```
audio = []
sample_rate = 44100.0


def append_silence(duration_milliseconds=500):

num_samples = duration_milliseconds * (sample_rate / 1000.0)

for x in range(int(num_samples)):
audio.append(0.0)

return


def append_sinewave(
freq=440.0,
duration_milliseconds=500,


global audio

num_samples = duration_milliseconds * (sample_rate / 1000.0)

for x in range(int(num_samples)):
audio.append(volume * math.sin(2 * math.pi * freq * ( x / sample_rate )))

return


def save_wav(file_name):

wav_file=wave.open(file_name,"w")
nchannels = 1

sampwidth = 2


nframes = len(audio)
comptype = "NONE"
compname = "not compressed"
wav_file.setparams((nchannels, sampwidth, sample_rate, nframes, comptype, compname))

for sample in audio:
wav_file.writeframes(struct.pack('h', int( sample * 32767.0 )))

wav_file.close()

return
```

```
append_sinewave(volume=0.25)
append_silence()
append_sinewave(volume=0.5)
append_silence()
append_sinewave()
save_wav("output.wav")

sound = "output.wav"
fs, sound = wavfile.read(sound)
plt.plot(sound)
plt.title("Signal Time", size=0.1)
n=len(sound)
freq=fft(sound)
freq = freq[0:int(np.ceil((n+1)/2.0))]
mg = np.abs(sound)
mg = mg/float(n)
mg = mg**2
if n%2 >0:
mg[1:len(freq)] = mg[1:len(mg)] * 2
else:
mg[1:len(mg) - 1] = mg[1:len(mg)-1]*2

plt.figure()
axis = np.arange(0,int(np.ceil((n+1)/2.0)),1.0) * (fs/n)
plt.plot(axis,axis)
plt.xlabel('Frequency')
plt.ylabel('Spectrum')
```





3a)tile1a

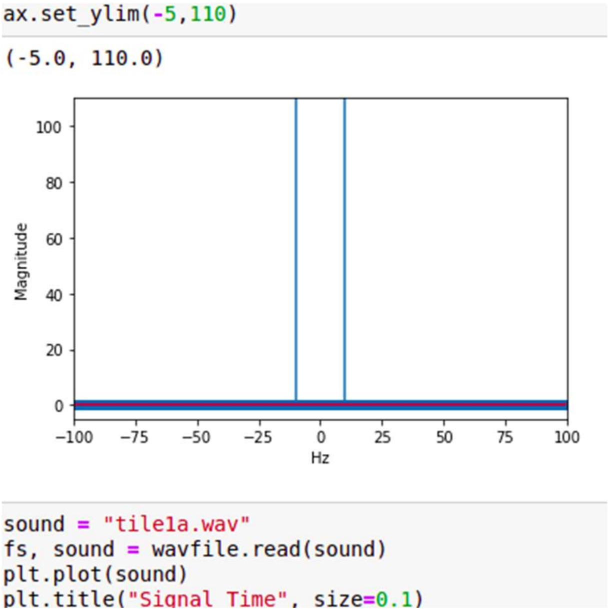


```
x.set_ylabel('Magnitude')
x.set_xlim(-N/2,N/2)
x.set_ylim(-5,110)
```

```
-5.0, 110.0)
```

```
amplerate, data = wavfile.read('cos_1khz_pulse_20msec
```

tile2a

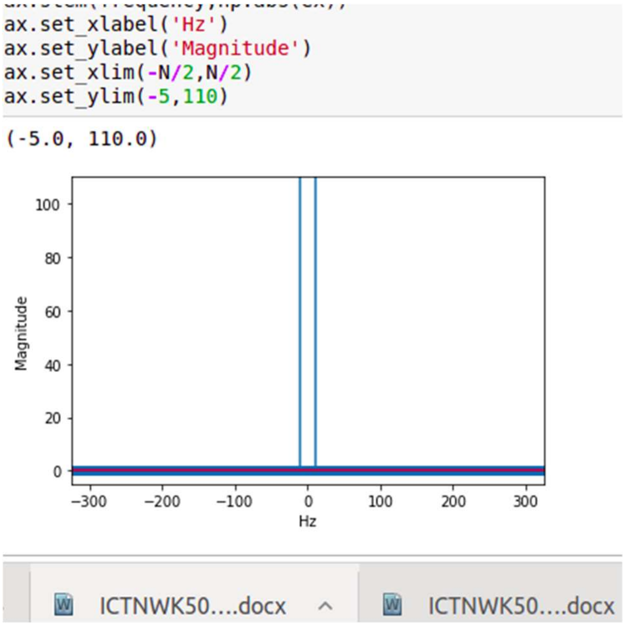


```
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

```
sound = "tile1a.wav"
fs, sound = wavfile.read(sound)
plt.plot(sound)
plt.title("Signal Time", size=0.1)
```

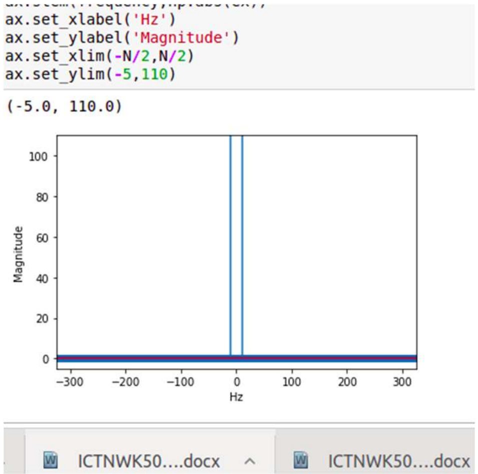
tile1b



```
ax.stem(frequency,np.abs(cx))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

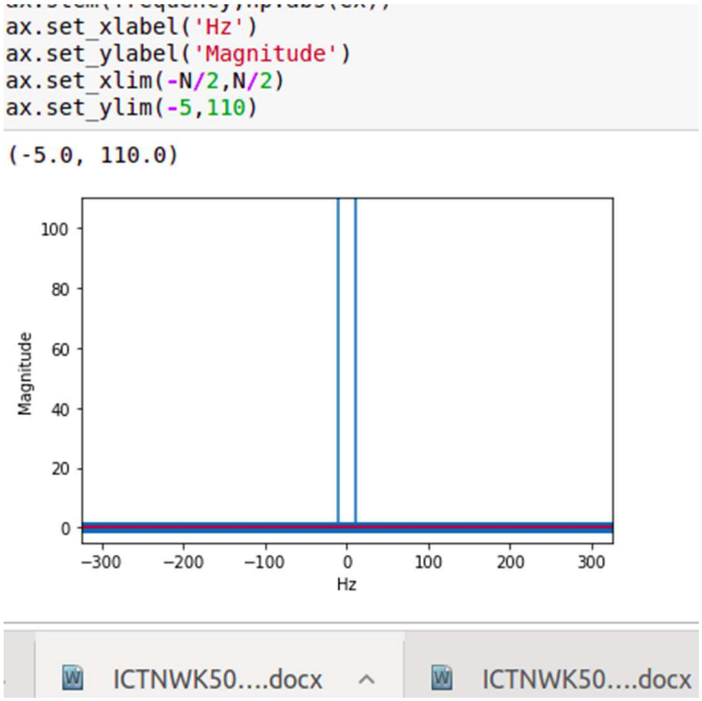
tile 2b



```
ax.stem(frequency,np.abs(cx))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

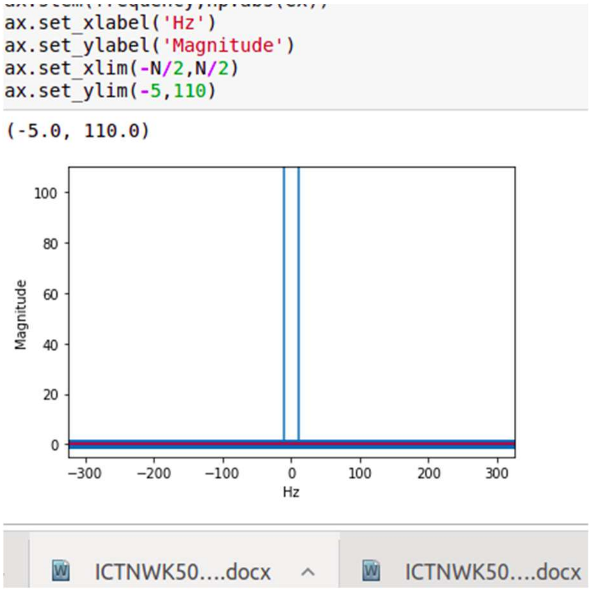
tile 1c



```
ax.stem(frequency,np.abs(cx))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

tile2c

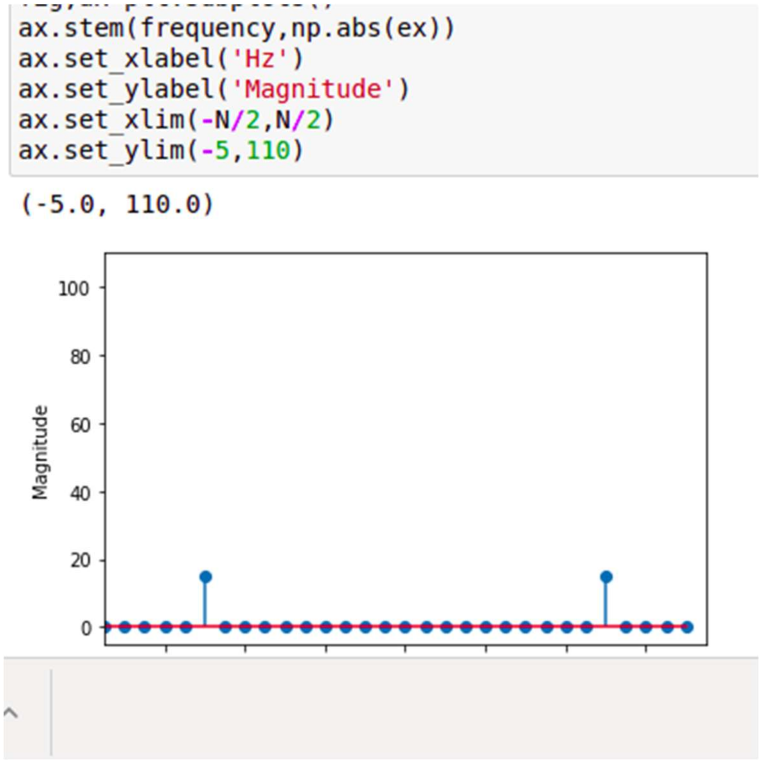


```
ax.stem(frequency,np.abs(cx))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

tile1d

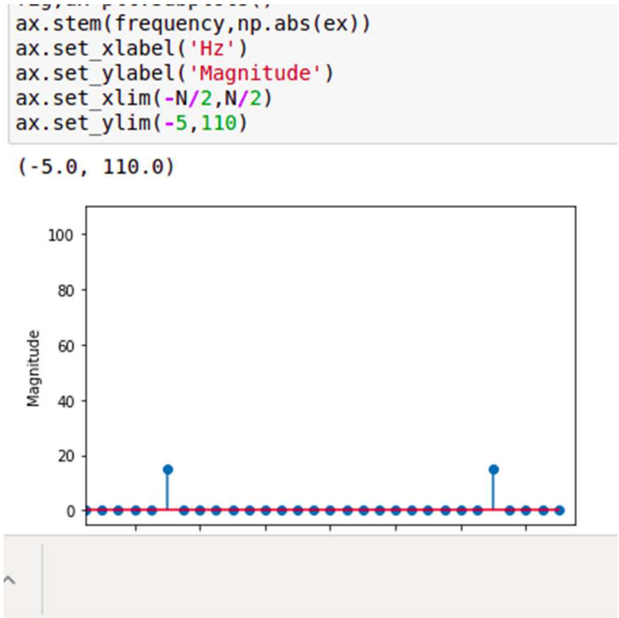
‘



```
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

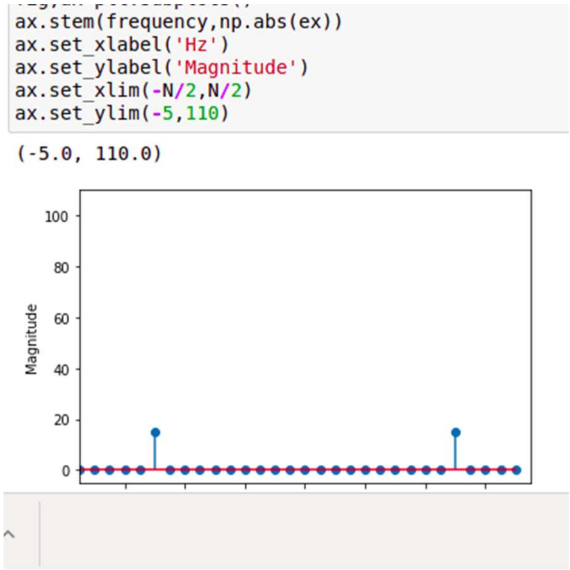
tile2d



```
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

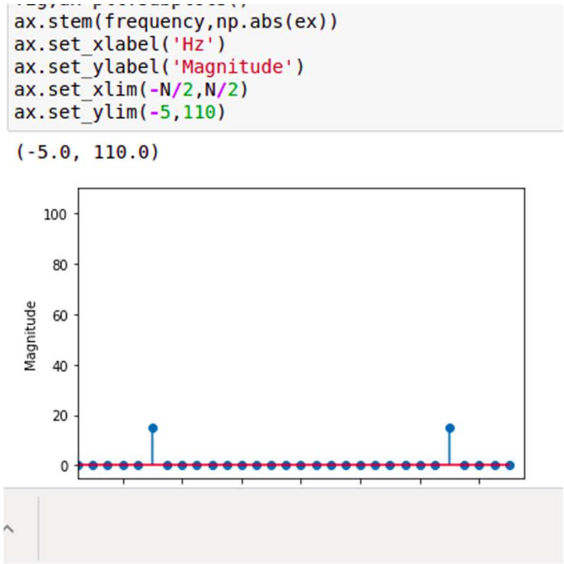
tile1e



```
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

tile2e

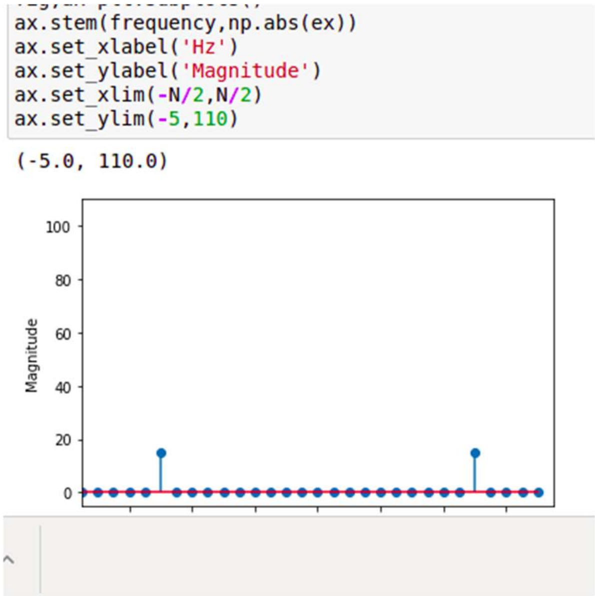


```
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

3b)

The frequency varies from -1000 to 1000. But is fully distorted. This audio file has issues



```
ax.stem(frequency,np.abs(ex))
ax.set_xlabel('Hz')
ax.set_ylabel('Magnitude')
ax.set_xlim(-N/2,N/2)
ax.set_ylim(-5,110)
```

```
(-5.0, 110.0)
```

3c) if n%2 >0:

mg[1:len(freq)] = mg[1:len(mg)] * 2
else:
mg[1:len(mg) - 1] = mg[1:len(mg)-1]*2

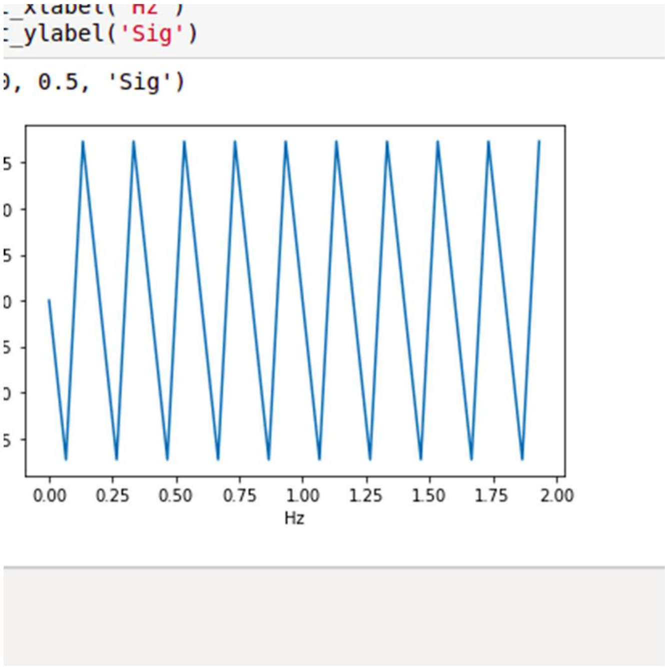
plt.figure()
axis = np.arange(0,int(np.ceil((n+1)/2.0)),1.0) * (fs/n)
plt.plot(axis,axis)
plt.xlabel('Frequency')
plt.ylabel('Spectrum')

3d)

a= 100

```
N = 200
t = np.linspace(0,2,2 * N, endpoint=False)
x = np.sin(f*2*np.pi*t)
fig,ax=plt.subplots()
ax.plot(t,x)
ax.set_xlabel('Hz')
ax.set_ylabel('Sig')
```

Frequency is showing a lot of variation. Can not say a particular value to 0.00 to 2.00



```
:_xlabel('Hz')
:_ylabel('Sig')
```

```
), 0.5, 'Sig')
```