```
#line one
#install ffmpeg
!pip install ffmpeg-python

#line two
#import libraries and define function
from IPython.display import HTML, Audio
from google.colab.output import eval_js
from base64 import b64decode
import numpy as np
from scipy.io.wavfile import read as wav_read
import io
import ffmpeg

AUDIO_HTML = """
<script>
var my_div = document.createElement("DIV");
var my_p = document.createElement("P");
var my_btn = document.createElement("BUTTON");
var t = document.createTextNode("Press to start recording");

my_btn.appendChild(t);
//my_p.appendChild(my_btn);
my_div.appendChild(my_btn);
document.body.appendChild(my_div);

var base64data = 0;
var reader;
var recorder, gumStream;
var recordButton = my_btn;

var handleSuccess = function(stream) {
gumStream = stream;
var options = {
//bitsPerSecond: 8000, //chrome seems to ignore, always 48k
mimeType : 'audio/webm;codecs=opus'
//mimeType : 'audio/webm;codecs=pcm'
};
//recorder = new MediaRecorder(stream, options);
recorder = new MediaRecorder(stream);
recorder.ondataavailable = function(e) {
var url = URL.createObjectURL(e.data);
var preview = document.createElement('audio');
preview.controls = true;
preview.src = url;
document.body.appendChild(preview);

reader = new FileReader();
reader.readAsDataURL(e.data);
reader.onloadend = function() {
base64data = reader.result;
//console.log("Inside FileReader:" + base64data);
}
};
recorder.start();
};

recordButton.innerText = "Recording... press to stop";

navigator.mediaDevices.getUserMedia({audio: true}).then(handleSuccess);


function toggleRecording() {
if (recorder && recorder.state == "recording") {
recorder.stop();
gumStream.getAudioTracks()[0].stop();
recordButton.innerText = "Saving the recording... pls wait!"
}
}

// https://stackoverflow.com/a/951057
function sleep(ms) {
```

```
return new Promise(resolve => setTimeout(resolve, ms));
}

var data = new Promise(resolve=>{
//recordButton.addEventListener("click", toggleRecording);
recordButton.onclick = ()=>{
toggleRecording()

sleep(2000).then(() => {
// wait 2000ms for the data to be available...
// ideally this should use something like await...
//console.log("Inside data:" + base64data)
resolve(base64data.toString())

});

}
});

</script>
"""

def get_audio():
display(HTML(AUDIO_HTML))
data = eval_js("data")
binary = b64decode(data.split(',')[1])

process = (ffmpeg
.input('pipe:0')
.output('pipe:1', format='wav')
.run_async(pipe_stdin=True, pipe_stdout=True, pipe_stderr=True, quiet=True,
overwrite_output=True)
)
output, err = process.communicate(input=binary)

riff_chunk_size = len(output) - 8
# Break up the chunk size into four bytes, held in b.
q = riff_chunk_size
```

```
b = []
for i in range(4):
q, r = divmod(q, 256)
b.append(r)

# Replace bytes 4:8 in proc.stdout with the actual size of the RIFF chunk.
riff = output[:4] + bytes(b) + output[8:]

sr, audio = wav_read(io.BytesIO(riff))

return audio, sr


#line three
#call function
audio, Fs = get_audio()

#line Four
audio = audio[:, 0] #selct only first list which has data

#line Five
#plot tin time domain
import matplotlib.pyplot as plt
plt.figure(figsize=(20,10))
plt.plot(audio)
plt.show()


#line six
#plot magnitude spectrum
plt.figure(figsize=(20,10))
plt.magnitude_spectrum(audio, sr)
plt.title('Magnitude spectrum')
```