Análisis del archivo de audio en formato WAV.

Un formato de audio muy difundido es el WAV. A continuación te ejemplificamos como cargar un archivo WAV para que puedan procesar las amplitudes del archivo en el tiempo.

Para cargar el archivo desde su PC pueden usar el siguiente código. La variable `fname` guarda el nombre del archivo.

```
from google.colab import files
file = files.upload()
fname = next(iter(file))
```

También pueden cargar el archivo desde una URL externa. A continuación se proporciona una URL con un archivo de ejemplo para que verifiquen.

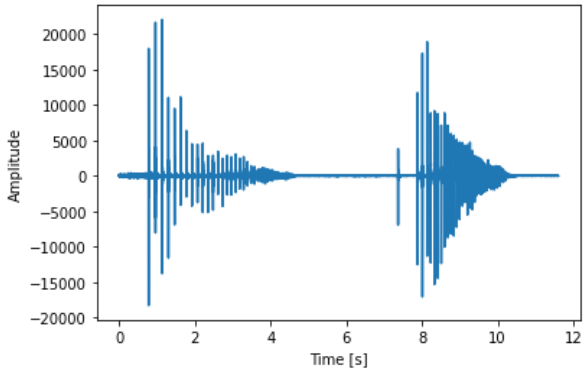
```
import requests
url= "https://dcubaar-my.sharepoint.com/:u:/g/personal/diegoshalom_dcubaar_onmicrosoft_com/EckkwZJ7QaRDruZqQnc9MMcB4x78
fname = "recording.wav"
downloaded_obj = requests.get(url)
with open(fname, "wb") as file:
    file.write(downloaded_obj.content)
```

En los siguientes bloques pueden: procesar el archivo WAV; graficar las amplitudes en el tiempo; buscar picos y máximos locales con una distancia mínima y una altura mínima.

```
from scipy.io import wavfile
import scipy.io

samplerate, x = wavfile.read(fname)
length = x.shape[0] / samplerate
```

```
import matplotlib.pyplot as plt
import numpy as np
time = np.linspace(0., length, x.shape[0])
plt.plot(time, x)
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud") # La amplitud está en unidades arbitrarias
plt.show()
```




```
from scipy.signal import find_peaks

distancia_minima = 500 # Ojo, está en samples, no en segundos. Si queremos una diferencia de DT seg sería: DT*samplerat
altura_minima = 1000   # No detectas picos menores a esto
peaks, _ = find_peaks(x, height=altura_minima ,distance=distancia_minima)

plt.plot(time,x)
plt.plot(time[peaks], x[peaks], "x")
plt.xlabel("Tiempo [s]")
```

```
plt.ylabel("Amplitud [u.a.]") # La amplitud está en unidades arbitrarias
plt.xlim(.5,5)
plt.show()
```