A simple trial

Fourier transform of n data points returns $n+1$ complex numbers $z_n = a_n + b_n i$ where a_n are the the $\cos(x)$ contributions in the signal and b_n are the the $\sin(x)$ contributions in the signal.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('./sample_data/sample2_out.csv')
print(df)
```

```
                Re        Im
0       8.389450  0.000000
1       5.105280 -1.901240
2       5.068870  1.891270
3       8.381400  0.038920
4       5.125340 -1.912640
...          ...       ...
122437 -0.007879  0.009161
122438  0.008523  0.000975
122439 -0.011258 -0.007286
122440 -0.005873  0.006421
122441  0.005192  0.000000

[122442 rows x 2 columns]
```

Audio specific data

```
L = 244882 # Length of data (n)
sampRate = 44100; # hz
```

Calculating frequency spectrum

The magnitude of sound is calculated as

$M_n = \sqrt{Re(z_n)^2 + Im(z_n)^2} = \sqrt{a_n^2 + b_n^2}.$

We can calculate this for each n and store those values in a new column of the dataframe

```
df["Mag"] = np.sqrt(df["Re"]*df["Re"] + df["Im"]*df["Im"])
print(df)
```

```
                Re        Im       Mag
0       8.389450  0.000000  8.389450
1       5.105280 -1.901240  5.447807
2       5.068870  1.891270  5.410208
3       8.381400  0.038920  8.381490
4       5.125340 -1.912640  5.470585
...          ...       ...       ...
122437 -0.007879  0.009161  0.012083
122438  0.008523  0.000975  0.008578
122439 -0.011258 -0.007286  0.013410
122440 -0.005873  0.006421  0.008702
122441  0.005192  0.000000  0.005192

[122442 rows x 3 columns]
```

To find the frequencies associated with these magnitudes, we use the following fourier specific calculation:

$f = \text{linspace}(0, \frac{1}{2T}, \text{numpoints}),$

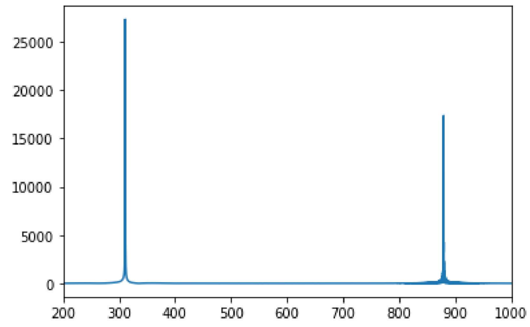
where numpoints is the number of data points in our sample and period T is the inverse of our sampling rate.

```
numpoints = int(L/2)
T = 1.0/sampRate
```

```
f = np.linspace(0, 1/(2*T), numpoints)
f = np.insert(f, 1, 0) # resize
df["Frequency"] = f.tolist()

print(df)
```

```
                Re        Im       Mag     Frequency
0       8.389450  0.000000  8.389450      0.000000
1       5.105280 -1.901240  5.447807      0.000000
2       5.068870  1.891270  5.410208      0.180088
3       8.381400  0.038920  8.381490      0.360176
4       5.125340 -1.912640  5.470585      0.540265
...          ...       ...       ...           ...
122437 -0.007879  0.009161  0.012083  22049.279647
122438  0.008523  0.000975  0.008578  22049.459735
122439 -0.011258 -0.007286  0.013410  22049.639824
122440 -0.005873  0.006421  0.008702  22049.819912
122441  0.005192  0.000000  0.005192  22050.000000

[122442 rows x 4 columns]
```

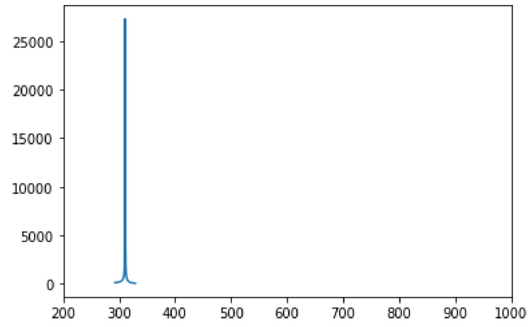
```
# plot
fig, ax = plt.subplots()

ax.plot(df["Frequency"], df["Mag"]); # they start at the same place so maybe clip off the last bit of data?
plt.xlim(200, 1000) # x limits
plt.show();
```



The parsing process

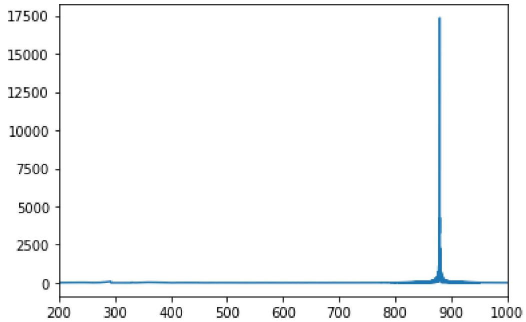
We isolate the first frequency by locating a peak in the magnitude data and grabbing a window of data around the peak

```
ind = 1728 # highest index
lowerbound = ind - 100;
upperbound = ind + 100;


# plot
fig, ax = plt.subplots()


ax.plot(df["Frequency"][lowerbound:upperbound], df["Mag"][lowerbound:upperbound]); # they start at the same place so maybe clip off the last
plt.xlim(200, 1000) # x limits
plt.show();
```



The second peak is calculated by removing the first peak -- setting the magnitude data within the selected index window to 0

```
# Kill the already isolated peak


df["Mag"][lowerbound:upperbound] = 0;


# plot
fig, ax = plt.subplots()

ax.plot(df["Frequency"], df["Mag"]);
plt.xlim(200, 1000) # x limits
plt.show();
```



What about a more legit audio?

```
df2 = pd.read_csv('./sample_data/jingle_out.csv')
print(df2)
```

```
                 Re         Im
0       108.723000   0.000000
1       -13.533700   4.356070
2         6.828180  24.743000
3       -18.943900 -12.546300
4        21.483300  10.820500
...            ...        ...
162933   -0.006334  -0.001345
162934    0.020293  -0.002687
162935    0.010724   0.005754
162936   -0.034264  -0.006255
162937   -0.021061   0.000000

[162938 rows x 2 columns]
```

```
L_jingle = 325874 # Length of data (n)
sampRate_jingle = 44100; # hz
```

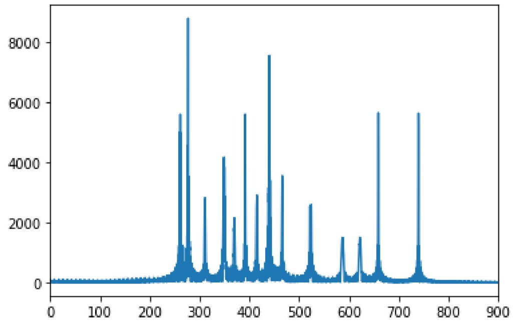
```
df2["Mag"] = np.sqrt(df2["Re"]*df2["Re"] + df2["Im"]*df2["Im"])
print(df2)
```

```
                 Re         Im         Mag
0       108.723000   0.000000  108.723000
1       -13.533700   4.356070   14.217467
2         6.828180  24.743000   25.667881
3       -18.943900 -12.546300   22.721818
4        21.483300  10.820500   24.054426
...            ...        ...         ...
162933   -0.006334  -0.001345    0.006476
162934    0.020293  -0.002687    0.020470
162935    0.010724   0.005754    0.012171
162936   -0.034264  -0.006255    0.034830
162937   -0.021061   0.000000    0.021061

[162938 rows x 3 columns]
```

```
numpoints_jingle = int(L_jingle/2)
T_jingle = 1.0/sampRate_jingle
f_jingle = np.linspace(0, 1/(2*T_jingle), numpoints_jingle)
f_jingle = np.insert(f_jingle, 1, 0) # resize
df2["Frequency"] = f_jingle.tolist()

print(df2)
```

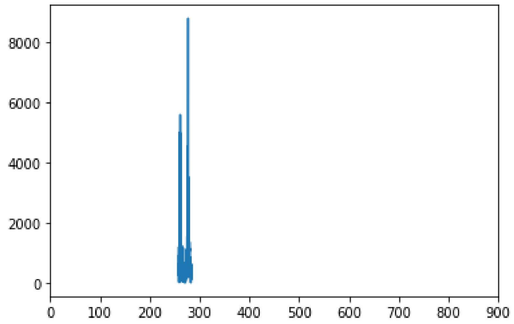
```
                Re         Im         Mag     Frequency
0       108.723000   0.000000  108.723000      0.000000
1       -13.533700   4.356070   14.217467      0.000000
2         6.828180  24.743000   25.667881      0.135329
3       -18.943900 -12.546300   22.721818      0.270658
4        21.483300  10.820500   24.054426      0.405988
...            ...        ...         ...           ...
162933   -0.006334  -0.001345    0.006476  22049.458683
162934    0.020293  -0.002687    0.020470  22049.594012
162935    0.010724   0.005754    0.012171  22049.729342
162936   -0.034264  -0.006255    0.034830  22049.864671
162937   -0.021061   0.000000    0.021061  22050.000000

[162938 rows x 4 columns]
```

```
# plot
fig, ax = plt.subplots()

ax.plot(df2["Frequency"], df2["Mag"]); # they start at the same place so maybe clip off the last bit of data?
plt.xlim(0, 900) # x limits
plt.show();
```



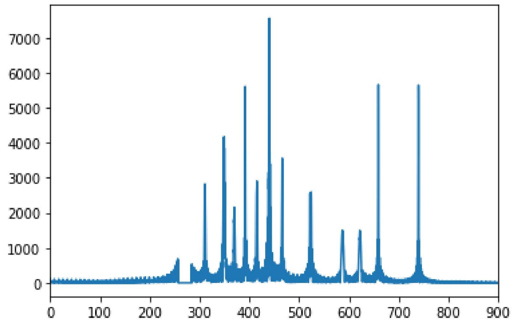
Parsing

```
ind = 2004 # highest index
lowerbound = ind - 100;
upperbound = ind + 100;


# plot
fig, ax = plt.subplots()


ax.plot(df2["Frequency"][lowerbound:upperbound], df2["Mag"][lowerbound:upperbound]);
plt.xlim(0, 900) # x limits
plt.show();
```



```
df2["Mag"][lowerbound:upperbound] = 0;


# plot
fig, ax = plt.subplots()
```

```
ax.plot(df2["Frequency"], df2["Mag"]); # they start at the same place so maybe clip off the last bit of data?
plt.xlim(0, 900) # x limits
plt.show();
```



✓ 0s completed at 9:48 AM