```
import os
import math
import random
import pickle
import itertools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
```

```
root_path = 'gdrive/My Drive/'
```

```
# catenate df1 and df2 together
df1 = pd.read_csv("gdrive/My Drive/tseries/ecg/mitbih_train.csv", header=None)
df2 = pd.read_csv("gdrive/My Drive/tseries/ecg/mitbih_test.csv", header=None)
df = pd.concat([df1, df2], axis=0)

df.head()
```

	0	1	2	3	4	5	6	7	
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.0588
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.0740
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.0540
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.0662
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.1173

5 rows × 188 columns

```
# df.info()
```

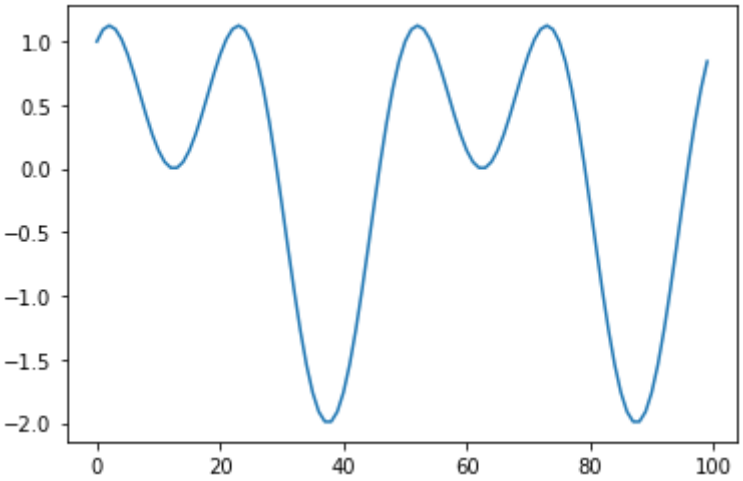
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109446 entries, 0 to 21891
Columns: 188 entries, 0 to 187
dtypes: float64(188)
memory usage: 157.8 MB
```

Fourier on Sine and Cosine Additive

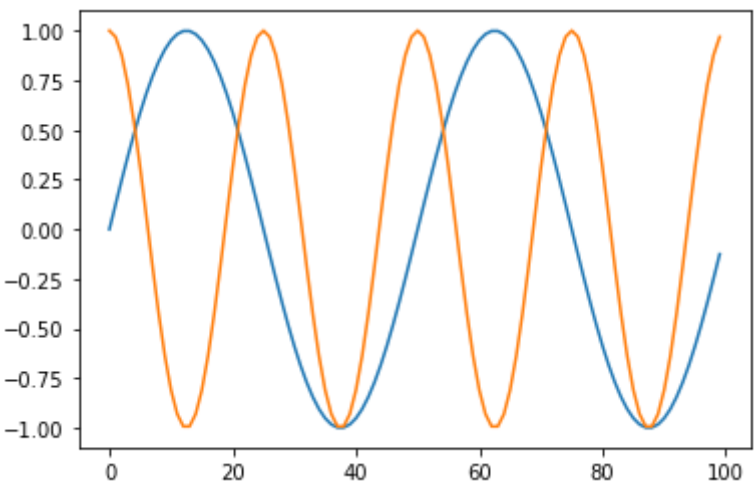
```
import numpy as np
import matplotlib.pyplot as plt
```

```
import matplotlib.pyplot as plt
T = 100
x = np.arange(0,T)
print(x)
y=  np.sin(4*np.pi*x/T)+np.cos(8*np.pi*x/T)
plt.plot(x, y)
plt.show()

print(y.shape)

sp   = np.fft.fft(y)                # the discrete fourier transform
freq = np.fft.fftfreq(y.shape[-1]) # the accompanying frequencies

cos=np.sum([(sp[-i]+sp[i]).real/(2*T)*np.cos(2.*np.pi*freq[i]*x)\
             for i in range(len(freq))],axis=0)
sin=np.sum([(sp[-i]-sp[i]).imag/200.*np.sin(2.*np.pi*freq[i]*x)\
              for i in range(len(freq))],axis=0)

plt.plot(x, sin)
plt.plot(x, cos)
plt.show()
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99]
```



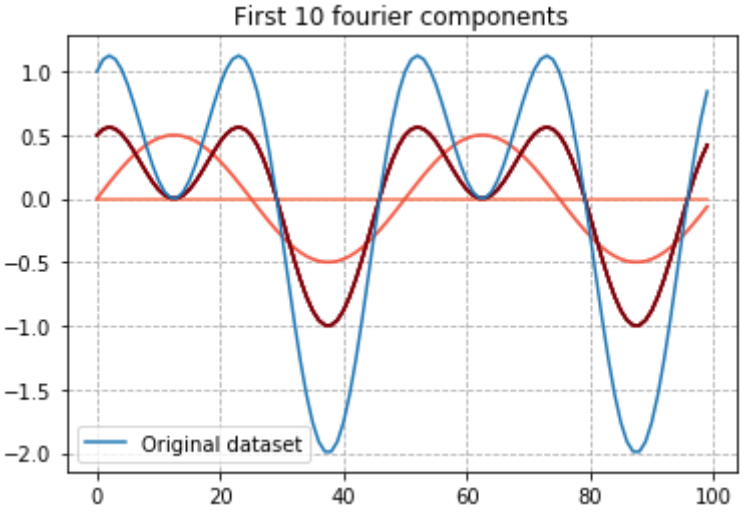

```
(100,)
```

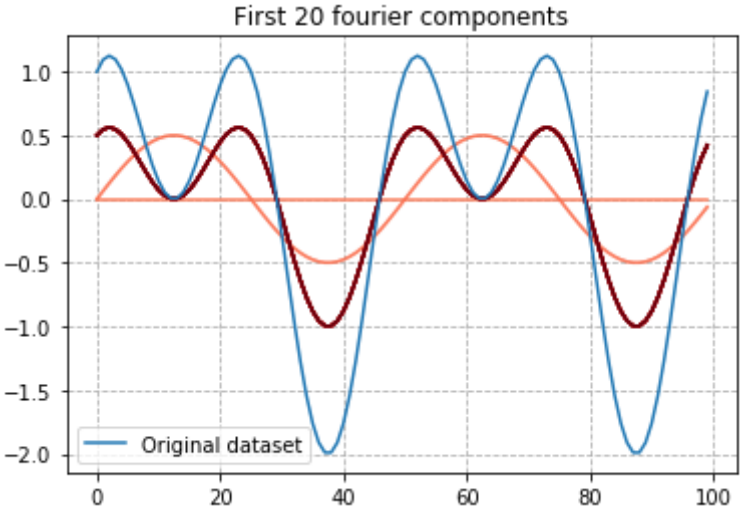


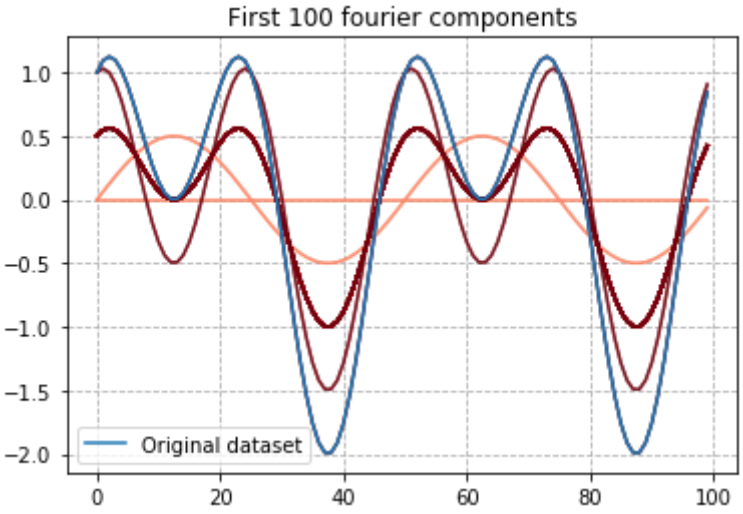
Iterative Component Split

```
import numpy
from matplotlib import pyplot as plt

n = len(y)
COMPONENTS = [10, 20, n]

for c in COMPONENTS:
    colors = numpy.linspace(start=100, stop=255, num=c)
    for i in range(c):
        Y = numpy.fft.fft(y)
        numpy.put(Y, range(i+1, n), 0.0)
        ifft = numpy.fft.ifft(Y)
        plt.plot(x, ifft, color=plt.cm.Reds(int(colors[i])), alpha=.70)

    plt.title("First {c} fourier components".format(c=c))
    plt.plot(x,y, label="Original dataset")
    plt.grid(linestyle='dashed')
    plt.legend()
    plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:85: ComplexWarni
  return array(a, dtype, copy=False, order=order)
```



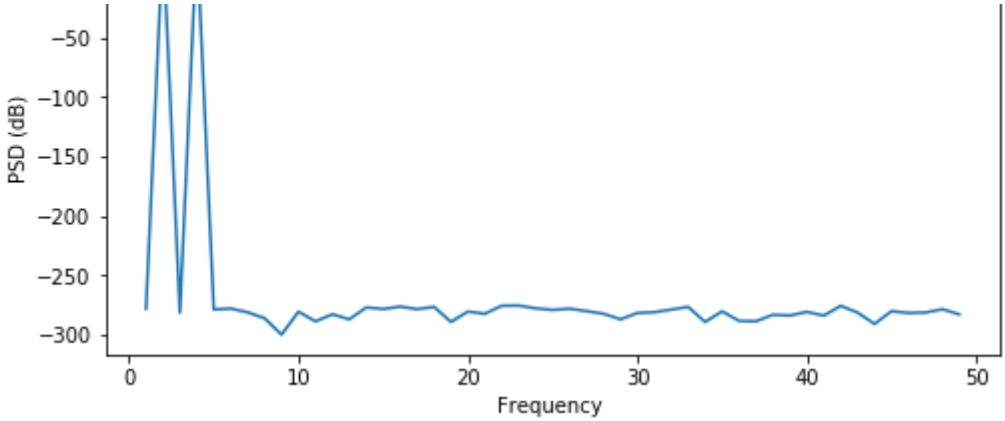


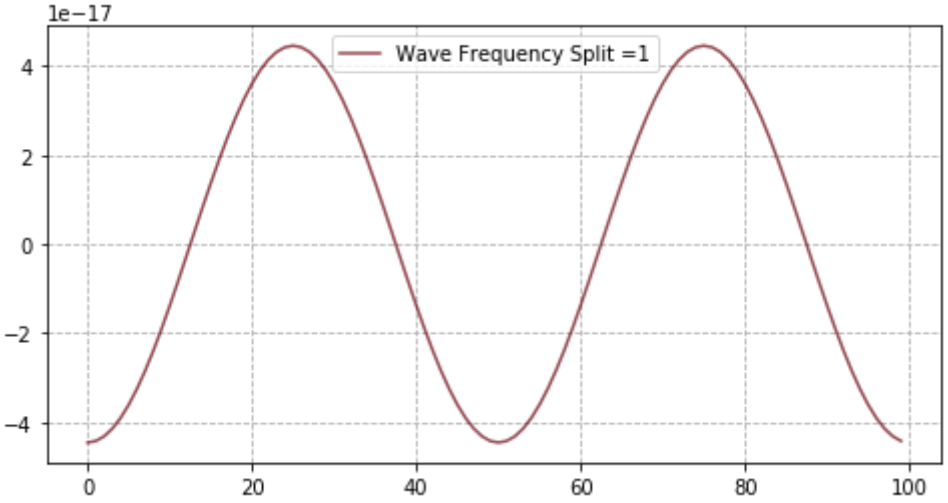


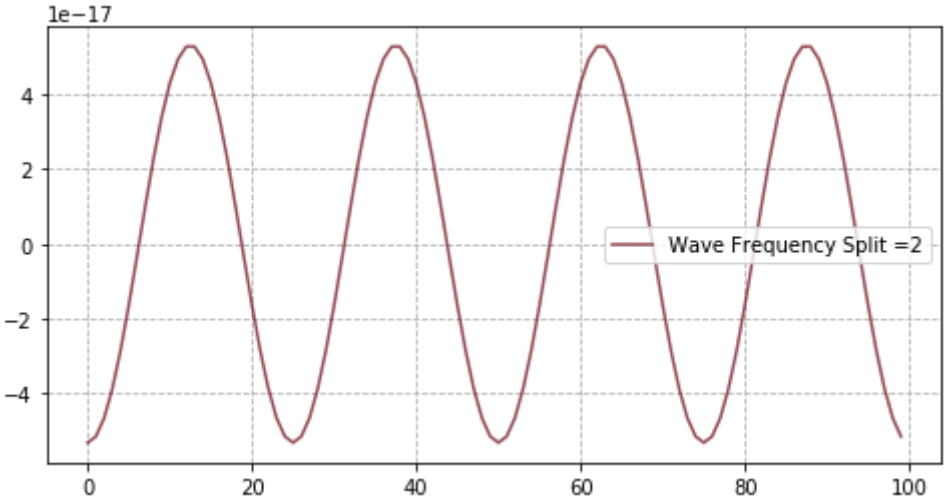
Fourier Decomposition of Component Waves

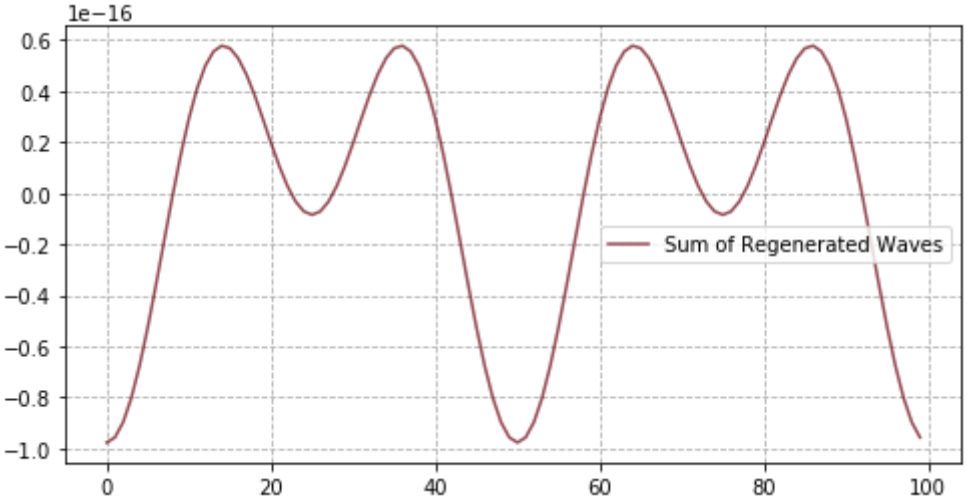
```
import datetime
import numpy as np
import scipy as sp
```

```
import scipy as sp
import scipy.fftpack
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks


plt.plot(x, y, color=plt.cm.Reds(int(colors[i])), alpha=.70)
plt.plot(x,y, label="Original dataset")
plt.grid(linestyle='dashed')
plt.legend()
plt.show()

n = len(y)
Y = numpy.fft.fft(y)

y_psd = np.abs(Y) ** 2
fftfreq = sp.fftpack.fftfreq(len(y_psd), 1. / 100)
pos = fftfreq > 0
fig, ax = plt.subplots(1, 1, figsize=(8, 4))
ax.plot(fftfreq[pos], 10 * np.log10(y_psd[pos]), label= "Power Spectral Density")
ax.set_xlabel('Frequency ')
ax.set_ylabel('PSD (dB)')

peaks, properties = find_peaks(10 * np.log10(y_psd[pos]))
# Visualize the first 2 peaks
ax.plot(fftfreq[pos][peaks][:2], 10 * np.log10(y_psd[pos])[peaks][:2], "x")

inv_fft_sum = np.zeros(len(Y))

count = 1
for p in peaks:

    if (p <= 3): #Peaks are 1 and 3
      temp_fft = np.zeros(len(Y))
      temp_fft[int(fftfreq[pos][p])] = Y[p]

      inv_fft = np.real(sp.fftpack.ifft(temp_fft))
      inv_fft_sum = inv_fft_sum + inv_fft

      fig, ax = plt.subplots(1, 1, figsize=(8, 4))
      plt.plot(x, inv_fft, color=plt.cm.Reds(int(colors[i])),
                  alpha=.70, label= "Wave Frequency Split =" + str(count))
      plt.grid(linestyle='dashed')
      plt.legend()
      plt.show()
      count = count + 1

fig, ax = plt.subplots(1, 1, figsize=(8, 4))
plt.plot(x, inv_fft_sum, color=plt.cm.Reds(int(colors[i])), alpha=.70, label="Sum o
plt.grid(linestyle='dashed')
plt.legend()
plt.show()
```


PSD (dB)
Frequency


1e−17
Wave Frequency Split =1


1e−17
Wave Frequency Split =2


1e−16
Sum of Regenerated Waves