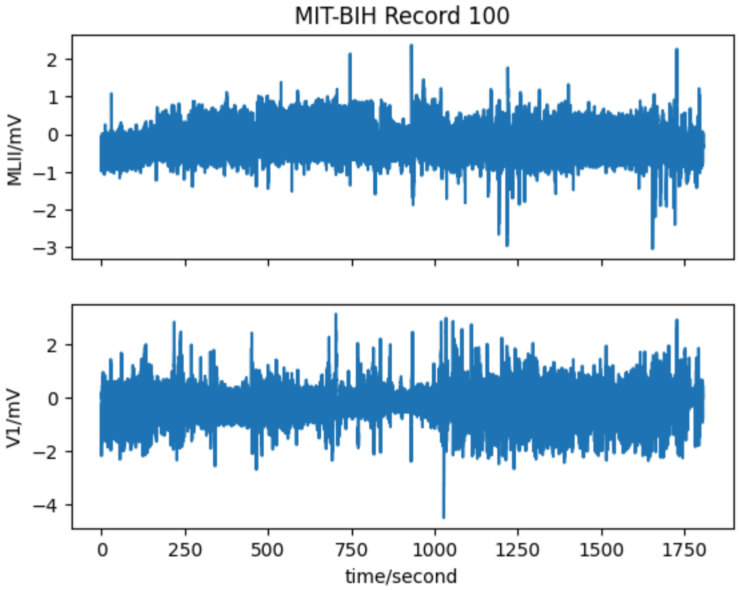
```
pip install wfdb
```

```
Collecting wfdb
  Downloading wfdb-4.3.0-py3-none-any.whl.metadata (3.8 kB)
Requirement already satisfied: aiohttp>=3.10.11 in /usr/local/lib/python3.11/dist-packages (from wfdb) (3.11.15)
Requirement already satisfied: fsspec>=2023.10.0 in /usr/local/lib/python3.11/dist-packages (from wfdb) (2025.3.2)
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.11/dist-packages (from wfdb) (3.10.0)
Requirement already satisfied: numpy>=1.26.4 in /usr/local/lib/python3.11/dist-packages (from wfdb) (2.0.2)
Collecting pandas>=2.2.3 (from wfdb)
  Downloading pandas-2.2.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (89 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.9/89.9 kB 1.6 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.8.1 in /usr/local/lib/python3.11/dist-packages (from wfdb) (2.32.3)
Requirement already satisfied: scipy>=1.13.0 in /usr/local/lib/python3.11/dist-packages (from wfdb) (1.15.2)
Requirement already satisfied: soundfile>=0.10.0 in /usr/local/lib/python3.11/dist-packages (from wfdb) (0.13.1)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wf
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wfdb) (1.
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wfdb) (25.3.
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wfdb) (1
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wfdb)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wfdb) (0.
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp>=3.10.11->wfdb) (1
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (0.12.
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (24
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb) (3
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.2.2->wfdb
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.2.3->wfdb) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.2.3->wfdb) (2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.8.1->wf
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.8.1->wfdb) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.8.1->wfdb) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.8.1->wfdb) (2
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.11/dist-packages (from soundfile>=0.10.0->wfdb) (1.17.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0->soundfile>=0.10.0->wfdb
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib>=3
Downloading wfdb-4.3.0-py3-none-any.whl (163 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 163.8/163.8 kB 3.5 MB/s eta 0:00:00
Downloading pandas-2.2.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 21.8 MB/s eta 0:00:00
Installing collected packages: pandas, wfdb
  Attempting uninstall: pandas
    Found existing installation: pandas 2.2.2
    Uninstalling pandas-2.2.2:
      Successfully uninstalled pandas-2.2.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is
google-colab 1.0.0 requires pandas==2.2.2, but you have pandas 2.2.3 which is incompatible.
Successfully installed pandas-2.2.3 wfdb-4.3.0
```

```
import numpy as np
import wfdb

# Read a sample ECG recording from the MIT-BIH Arrhythmia Database
record = wfdb.rdrecord('108', pn_dir='mitdb')

# Display the signal shape
print(record.p_signal.shape)

# Plot the ECG
wfdb.plot_wfdb(record=record, title='MIT-BIH Record 100')
```

```
(650000, 2)
```



```
record.sig_name
print(record.p_signal.shape)

display(signal)
display(fields)
```

```
(650000, 2)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-de1dc1c7939e> in <cell line: 0>()
      2 print(record.p_signal.shape)
      3 
----> 4 display(signal)
      5 display(fields)

NameError: name 'signal' is not defined
```

```
#signal, fields = wfdb.rdsamp('100', channels=[0],sampfrom=0, sampto=1000, pn_dir='mitdb')
signal, fields = wfdb.rdsamp('108', channels=[1], pn_dir='mitdb')
signal = signal.flatten()

print(signal.shape)
display(signal)
display(fields)
```

```
(650000,)
array([-0.78 , -0.78 , -0.78 , ...,  0.085,  0.08 ,  0.   ])
{'fs': 360,
 'sig_len': 650000,
 'n_sig': 1,
 'base_date': None,
 'base_time': None,
 'units': ['mV'],
 'sig_name': ['V1'],
 'comments': ['87 F 1227 654 x1',
  'Digoxin, Quinaglute',
  'There is borderline first degree AV block and sinus arrhythmia.  The',
  'PVCs are multiform.  The lower channel exhibits considerable noise and',
  'baseline shifts.']}
```

```
import scipy.io
scipy.io.savemat('/content/signal.mat', {'signal': signal})
```

```
# Demo 5 - Read a WFDB record and annotation. Plot all channels, and the annotation on top of channel 0.
fs=fields['fs']
```

```
time = np.arange(len(signal)) / fs
print(fs)
```

```
360
```

```
#ann=wfdb.rdann('108','atr',sampfrom=0, sampto=1000, pn_dir='mitdb')
ann=wfdb.rdann('108','atr', pn_dir='mitdb')
```

```
print(ann)
scipy.io.savemat('/content/ann.mat', {'ann': ann})
```

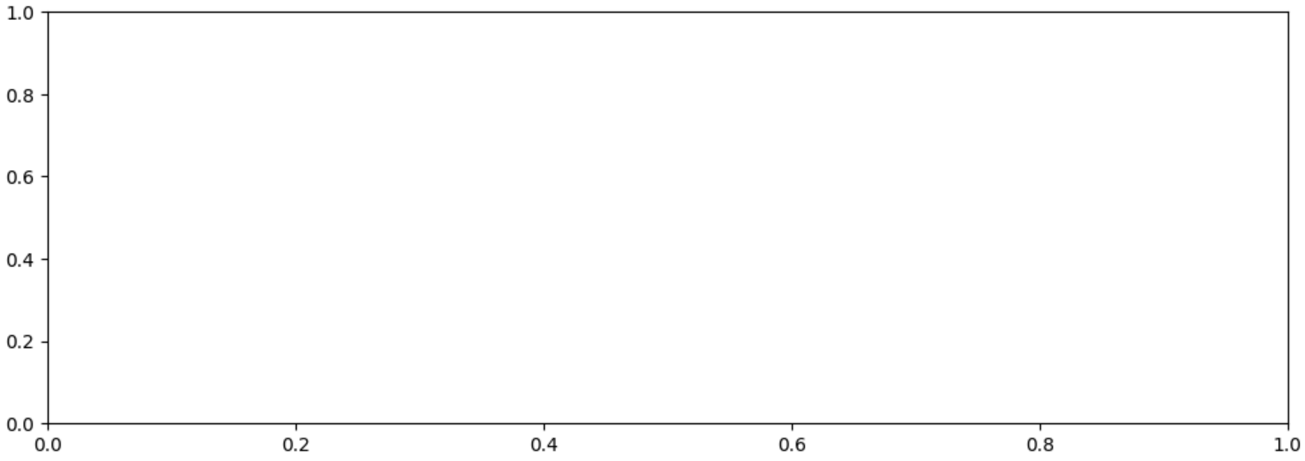
```
<wfdb.io.annotation.Annotation object at 0x789e85cf7cd0>
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-2ca0efcfa51e> in <cell line: 0>()
      1 print(ann)
----> 2 scipy.io.savemat('/content/ann.mat', {'ann': ann})

                                   6 frames

/usr/local/lib/python3.11/dist-packages/scipy/io/matlab/_mio5.py in write(self, arr)
    655         narr = to_writeable(arr)
    656         if narr is None:
--> 657             raise TypeError(f'Could not convert {arr} (type {type(arr)}) to array')
    658         if isinstance(narr, MatlabObject):
    659             self.write_object(narr)

TypeError: Could not convert None (type <class 'NoneType'>) to array
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))
plt.plot(time, signal, color='blue')
plt.plot(ann.sample/fs,signal[ann.sample],'ro')
plt.title('Channel 0 - Record 100 (MIT-BIH)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (mV)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-f5cc162879e9> in <cell line: 0>()
      1 import matplotlib.pyplot as plt
      2 plt.figure(figsize=(12, 4))
----> 3 plt.plot(time, signal, color='blue')
      4 plt.plot(ann.sample/fs,signal[ann.sample],'ro')
      5 plt.title('Channel 0 - Record 100 (MIT-BIH)')
```

3 frames

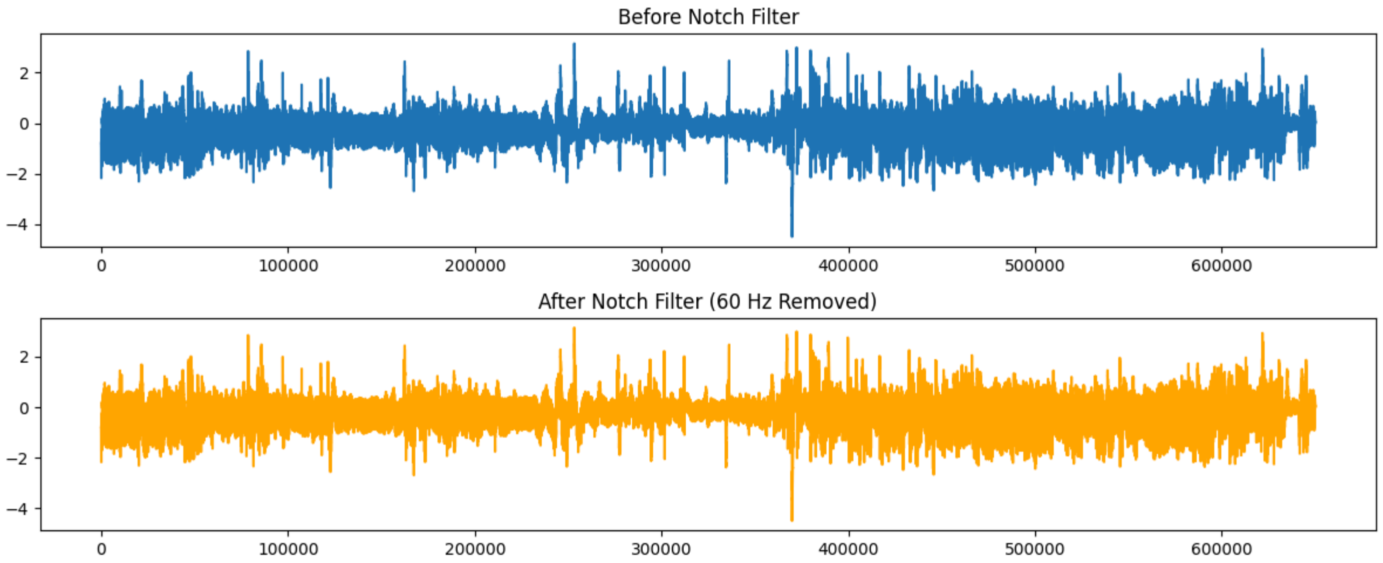
```
/usr/local/lib/python3.11/dist-packages/matplotlib/axes/_base.py in _plot_args(self, axes, tup, kwargs, return_kwargs,
ambiguous_fmt_datakey)
    492
    493         if x.shape[0] != y.shape[0]:
--> 494             raise ValueError(f"x and y must have same first dimension, but "
    495                              f"have shapes {x.shape} and {y.shape}")
    496         if x.ndim > 2 or y.ndim > 2:

ValueError: x and y must have same first dimension, but have shapes (650000,) and (1000,)
```



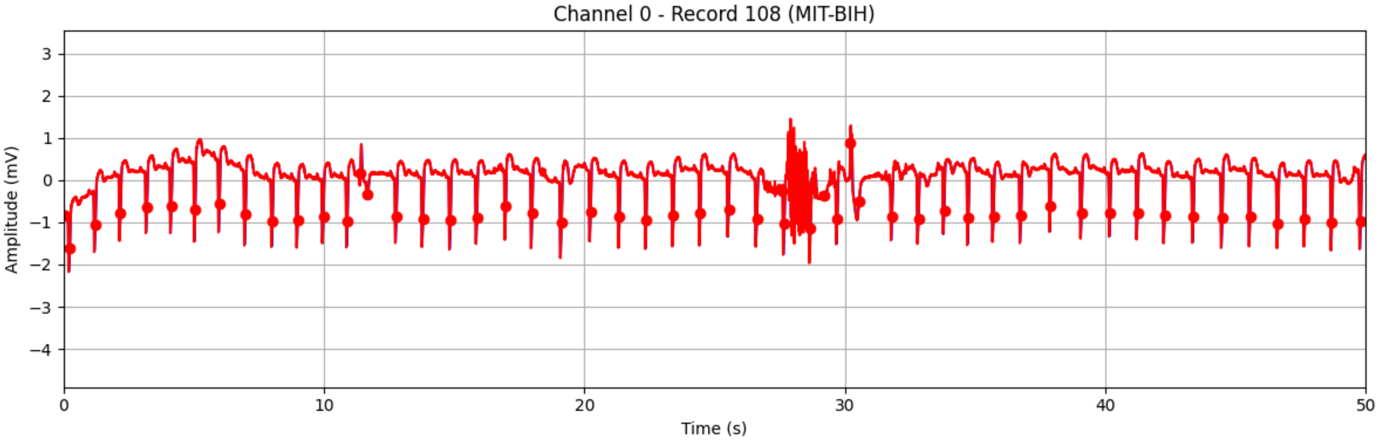
```
from scipy.signal import butter
#b, a = butter(N=2, Wn=0.33, btype='low', analog=False)  # fc=60Hz wn=fc/fs/2
b, a = butter(N=2, Wn=0.22, btype='low', analog=False)  # fc=40Hz wn=fc/fs/2
print(a)
```

```
[ 1.         -1.06224443  0.3786193 ]
```

```
from scipy.signal import iirnotch, filtfilt

fs = 360        # Sampling frequency (MITDB)
f0 = 60          # Frequency to remove
Q = 30           # Quality factor (higher = narrower notch)

# Design notch filter
wo = f0 / (fs/2)         # Normalized frequency
bw = wo / Q              # Bandwidth
[b, a] = iirnotch(wo, bw)

# Apply the filter
filtered = filtfilt(b, a, signal)  # Use filtfilt for zero-phase



# Optional: plot before and after
plt.figure(figsize=(12, 5))
plt.subplot(2, 1, 1)
plt.plot(signal, label='Original Signal')
plt.title('Before Notch Filter')
plt.subplot(2, 1, 2)
plt.plot(filtered, label='Filtered Signal', color='orange')
plt.title('After Notch Filter (60 Hz Removed)')
plt.tight_layout()
plt.show()
```

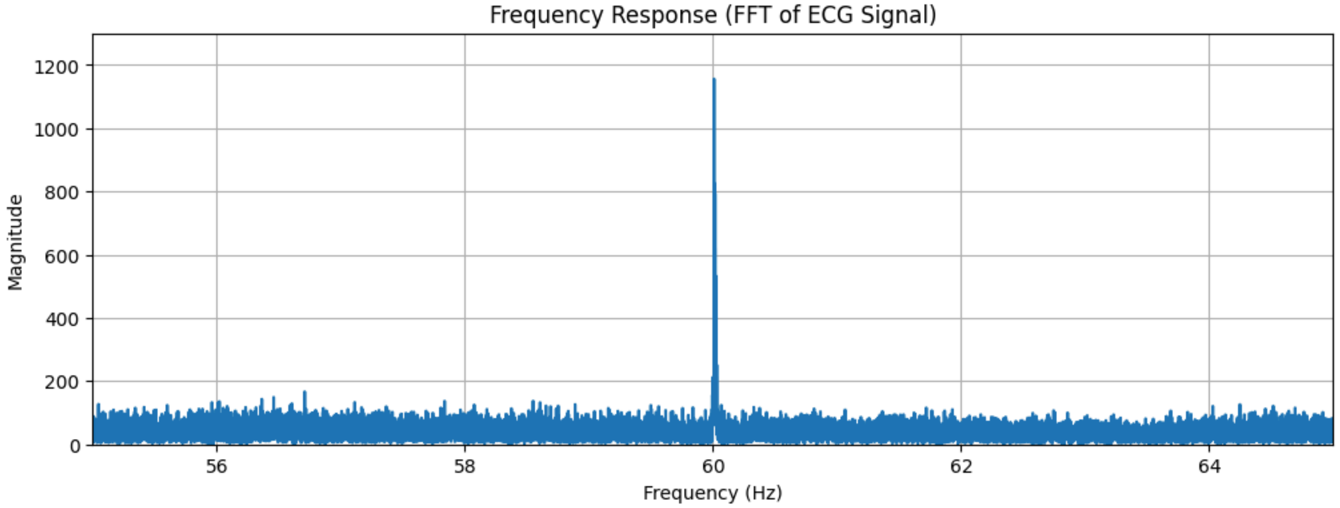


```
from scipy.signal import lfilter

filtered = lfilter(b, a, signal)



plt.figure(figsize=(12, 4))
plt.plot(time, filtered, color='blue')
plt.plot(ann.sample/fs,filtered[ann.sample],'ro')
plt.plot(time, signal, color='red')
plt.title('Channel 0 - Record 108 (MIT-BIH)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (mV)')
plt.xlim(0,50)
plt.grid(True)
plt.tight_layout()
plt.show()
```

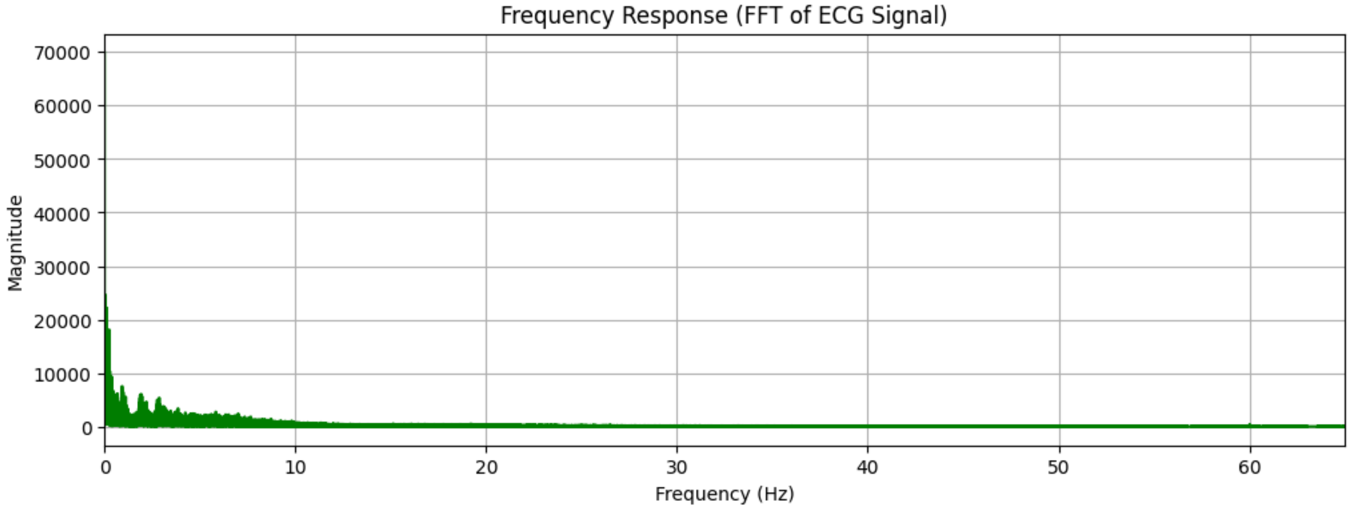


```
import numpy as np
T = 1 / fs  # Sampling interval
n = len(signal)
fft_vals = np.fft.fft(signal)
fft_vals = np.abs(fft_vals[:n // 2])  # Magnitude only, half-spectrum
freqs = np.fft.fftfreq(n, T)[:n // 2]  # Frequency bins



# Plot positive frequencies only
```

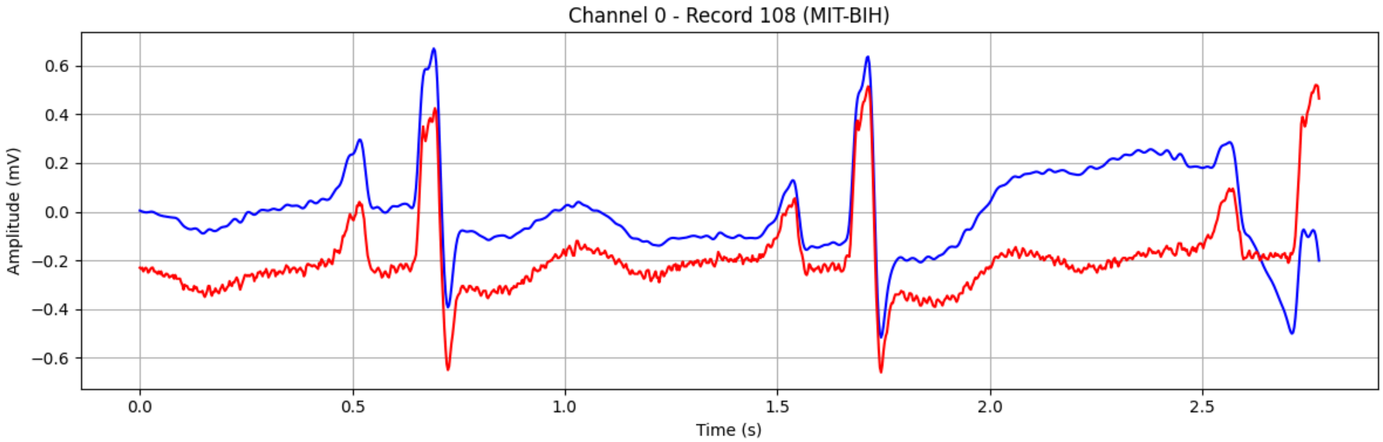
```
plt.figure(figsize=(12, 4))
plt.plot(freqs, fft_vals)
plt.title("Frequency Response (FFT of ECG Signal)")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(55,65)
plt.ylim(0,1300)
plt.grid(True)
plt.show()
```



```
fft_filt = np.fft.fft(filtered)
fft_filt = np.abs(fft_filt[:n // 2])  # Magnitude only, half-spectrum
freqs = np.fft.fftfreq(n, T)[:n // 2]  # Frequency bins



# Plot positive frequencies only
plt.figure(figsize=(12, 4))
plt.plot(freqs, fft_filt, color='green')
#plt.plot(freqs, fft_vals,color='red')
plt.title("Frequency Response (FFT of ECG Signal)")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 65)  # Zoom into 0 to 100 Hz


plt.grid(True)
plt.show()
```



```
from scipy.signal import butter
from scipy.signal import iirnotch, filtfilt
from scipy.signal import iirnotch, filtfilt
import matplotlib.pyplot as plt
for i in range(0,65000, 1000):
  signal, fields = wfdb.rdsamp('108', channels=[0], sampfrom=i, sampto=i+1000,pn_dir='mitdb')
  signal = signal.flatten()
  fs=fields['fs']
  time = np.arange(len(signal)) / fs
  filtered=signal_process(signal,fs)

print(i)
plt.figure(figsize=(12, 4))
plt.plot(time, filtered, color='blue')
plt.plot(time, signal, color='red')

plt.title('Channel 0 - Record 108 (MIT-BIH)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (mV)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

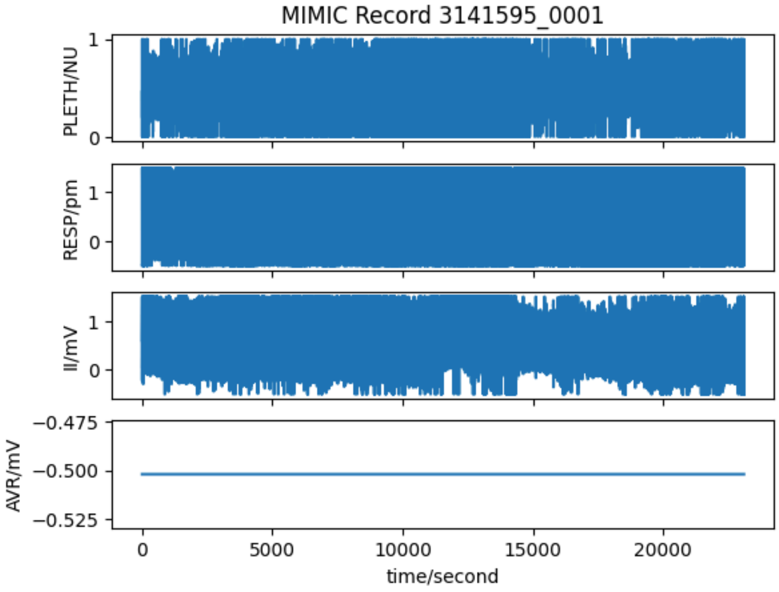


```
def signal_process (sig, fs,):
  low=0.5/(0.5*fs)
  high=40/(0.5*fs)
  b, a = butter(4, [low, high], btype='band', analog=False)
  filt = filtfilt(b, a, signal)  # Use filtfilt for zero-phase
  return filt
```

PPG

```
import numpy as np
import wfdb

record = wfdb.rdrecord('3141595_0001', pn_dir='mimic3wdb/1.0/31/3141595')



# Display the signal shape
print(record.p_signal.shape)

# Plot the ECG
wfdb.plot_wfdb(record=record, title='MIMIC Record 3141595_0001')
```

```
(2888500, 4)
```

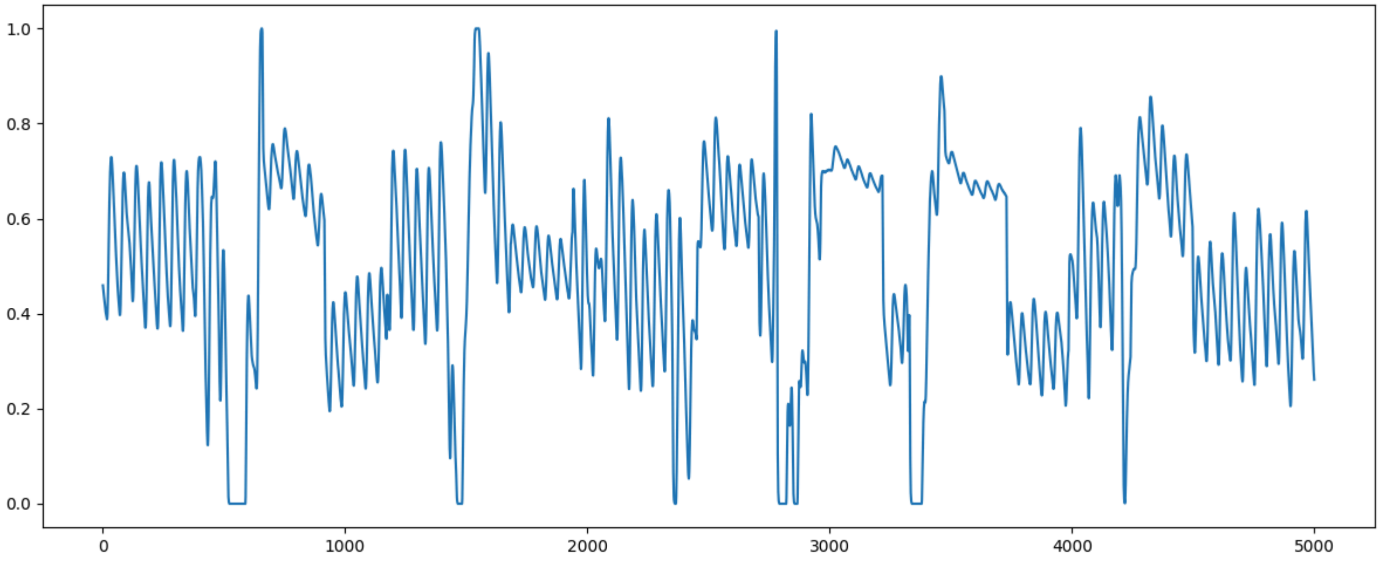


```
record.sig_name
print(record.p_signal.shape)
```

```
(2888500, 4)
```

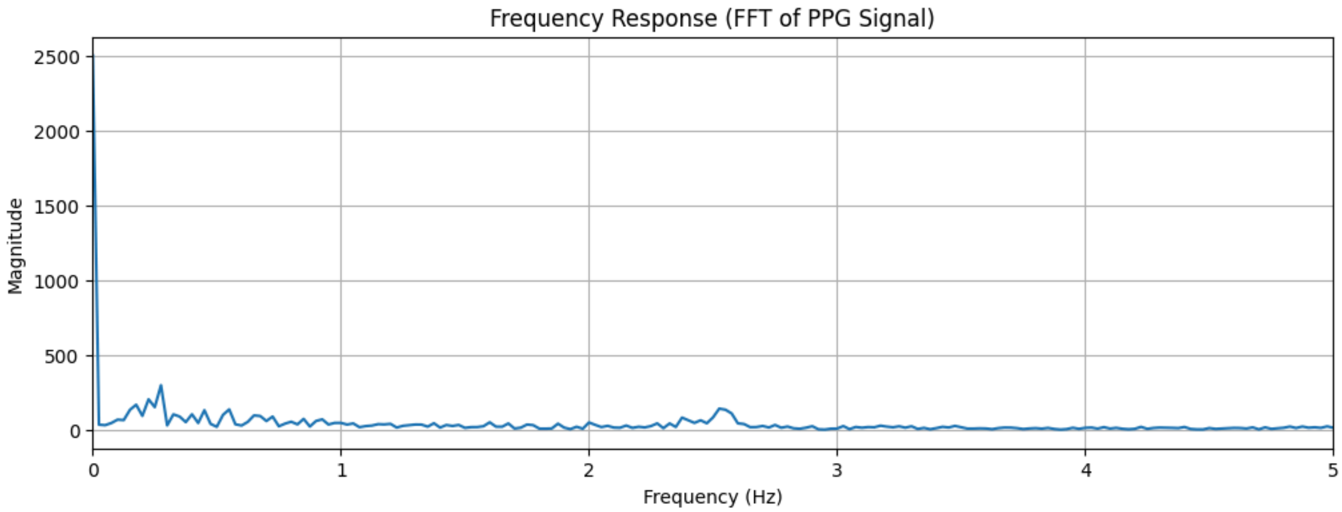
```
# Demo 5 - Read a WFDB record and annotation.
#signal, fields = wfdb.rdsamp('3141595_0001', sampfrom=0, sampto=1000, channels=[0], pn_dir='mimic3wdb/1.0/31/3141595')
signal, fields = wfdb.rdsamp('3141595_0001', sampfrom=0, sampto=5000, channels=[0], pn_dir='mimic3wdb/1.0/31/3141595')

signal = signal.flatten()
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))
plt.plot(signal, label='Original Signal')
plt.tight_layout()
plt.show()
```



```
T = 1 / fs  # Sampling interval
```

```
n = len(signal)
fft_vals = np.fft.fft(signal)
fft_vals = np.abs(fft_vals[:n // 2])  # Magnitude only, half-spectrum
freqs = np.fft.fftfreq(n, T)[:n // 2]  # Frequency bins



# Plot positive frequencies only
plt.figure(figsize=(12, 4))
plt.plot(freqs, fft_vals)
plt.title("Frequency Response (FFT of PPG Signal)")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0,5)
#plt.ylim(0,1300)
plt.grid(True)
plt.show()
```



```
print(signal.shape)
display(signal)
display(fields)
fs=fields['fs']
time = np.arange(len(signal)) / fs
print(fs)
```

```
(5000,)
array([0.45943304, 0.45356794, 0.44965787, ..., 0.27663734, 0.2688172 ,
       0.26099707])
{'fs': 125,
 'sig_len': 5000,
 'n_sig': 1,
 'base_date': None,
 'base_time': datetime.time(10, 2, 51, 840000),
 'units': ['NU'],
 'sig_name': ['PLETH'],
 'comments': []}
125
```

```
from scipy.signal import butter
low=0.5/(0.5*125)
high=5/(0.5*125)
b, a = butter(4, [low, high], btype='band', analog=False)
print(a)
```

```
[  1.          -7.38603474  23.90424922 -44.27951109  51.34936746
 -38.17591129  17.76963762  -4.73470781   0.55291062]
```

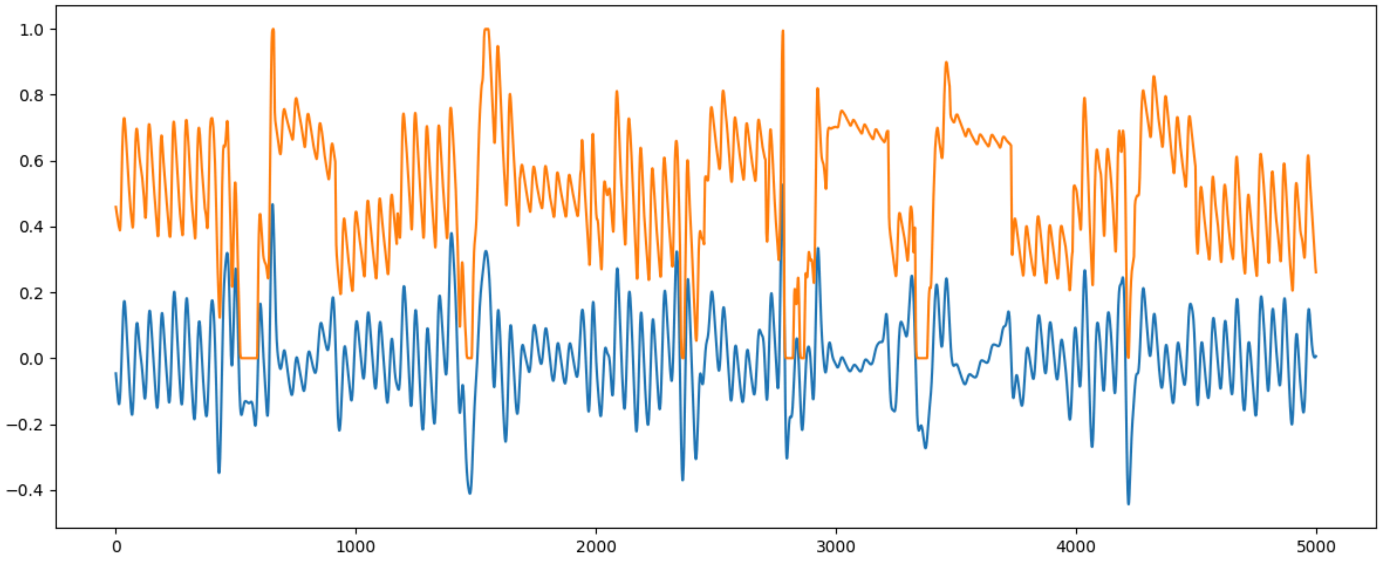
```
from scipy.signal import iirnotch, filtfilt
filtered = filtfilt(b, a, signal)  # Use filtfilt for zero-phase



b, a = butter(4, Wn=0.0064, btype='high', analog=False)
filtered = filtfilt(b, a, filtered)  # Use filtfilt for zero-phase
```

```
fft_filt = np.fft.fft(filtered)
fft_filt = np.abs(fft_filt[:n // 2])  # Magnitude only, half-spectrum
freqs = np.fft.fftfreq(n, T)[:n // 2]  # Frequency bins



# Plot positive frequencies only
plt.figure(figsize=(12, 4))
plt.plot(freqs, fft_filt, color='green')
#plt.plot(freqs, fft_vals,color='red')
plt.title("Frequency Response (FFT of PPG Signal)")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
#plt.xlim(0, 65)  # Zoom into 0 to 100 Hz


plt.grid(True)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-1961496625.py in <cell line: 0>()
----> 1 fft_filt = np.fft.fft(filtered)
      2 fft_filt = np.abs(fft_filt[:n // 2])  # Magnitude only, half-spectrum
      3 freqs = np.fft.fftfreq(n, T)[:n // 2]  # Frequency bins
      4
      5

NameError: name 'np' is not defined
```

Next steps: Explain error

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))
plt.plot(filtered, label='filtered Signal')
plt.plot(signal, label='Original Signal')

plt.tight_layout()
plt.show()
```



rough work from here onwards

```
record = wfdb.rdrecord('108', pn_dir='mitdb', sampto=1000)
sig = record.p_signal[:, 0]  # First channel

display(sig)
display(signal)
```