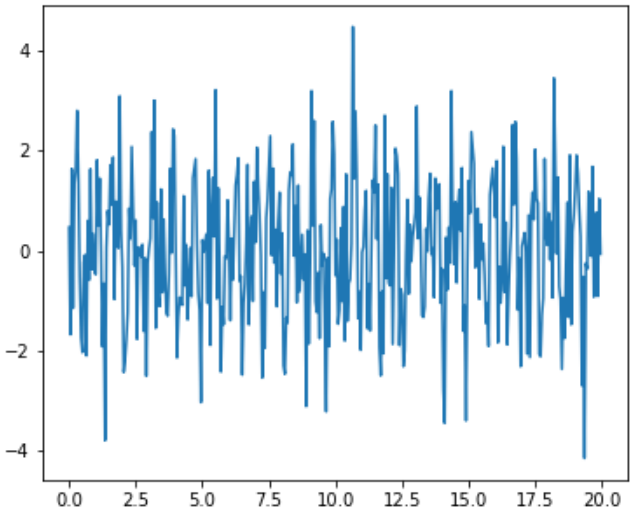
Lab 7
FFT

Task 1 - Noise Cancellation

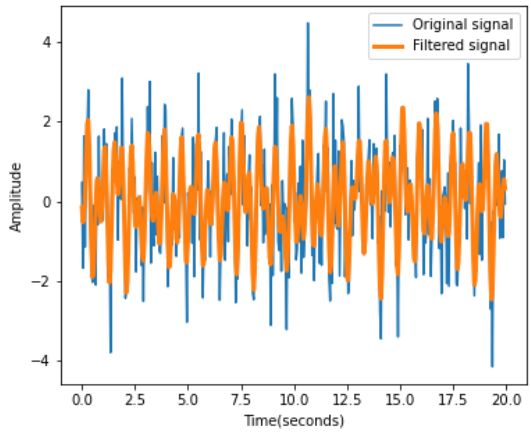
The first task is to create a filter, pass the signal through a filter, and then reconstruct the signal in the time domain. The original signal function and plot are shown below.

```
sig = (np.sin(8 * np.pi / period * time_vec) + 0.6 * np.random.randn(time_vec.size)) + (np.sin(105 * np.pi / period * time_vec) + 0.3 * np.random.randn(time_vec.size)) +(np.sin(550 * np.pi / period * time_vec)
```



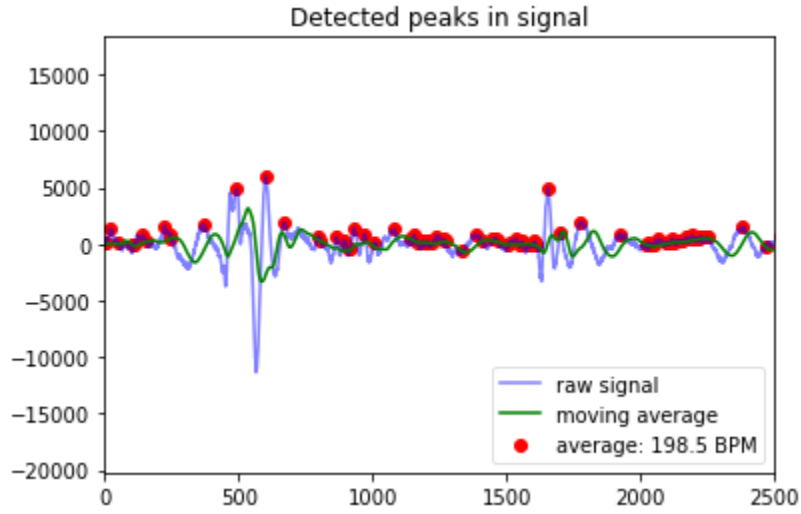
After creating the signal, we perform an FFT and plot the power to find the peak frequencies so the high frequencies can be found. Once high frequencies are identified, we can create a high frequency FFT filter. The original signal and the filtered signal are plotted over each other as shown below.

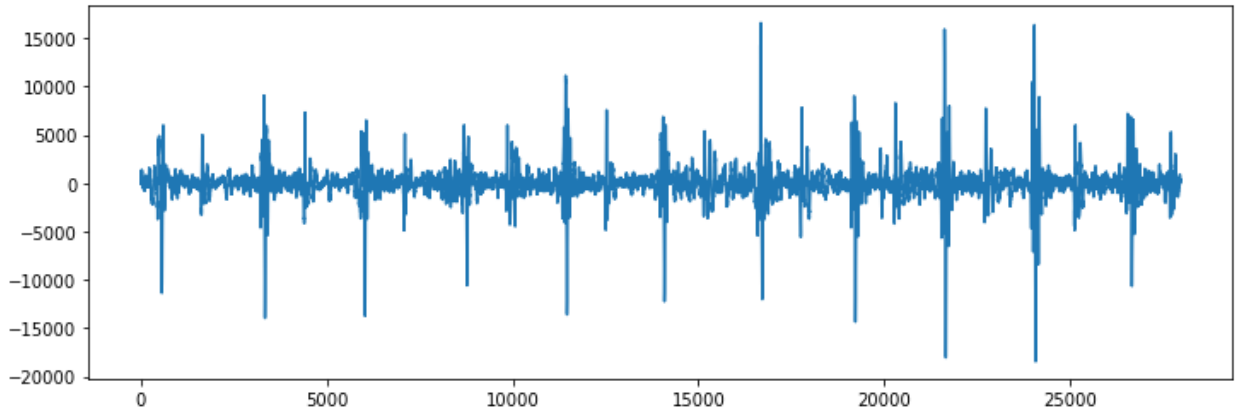
```
high_freq_fft = sig_fft.copy()
high_freq_fft[np.abs(sample_freq) > peak_freq] = 0
filtered_sig = fftpack.ifft(high_freq_fft)
```



Task 2 - Heartbeat Signal

A heartbeat signal was taken from the database found at https://www.kaggle.com/datasets/kinguistics/heartbeat-sounds?resource=download and converted to a csv format. To convert the wav file to a csv file, the wav2csv.py code was used. The wav file needs to be in a file called “wavfile” and within the same directory as the code. Once converted to a csv, the HeartRate.py code will plot the BPM and plot the time-domain form of the signal. The module “heartpy” needs to be installed for the code to work properly.





Task 3 - Red Star

In this game, the goal is to click a red snowflake before it reaches the bottom of the screen. As the levels increase, the number of stars will increase and descend faster. Once level 6 has been completed, a "Game Complete" screen will appear. If there is a misclick or the star reaches the bottom of the screen the game will restart and the player can try again. There were 3 changes made to this game: actor change (star to snowflake), speed of descent, and the ability to try the game again.

```
for color in colors_to_create:
    star = Actor("snowflake-" + color)
    new_stars.append(star)
```

Start to Snowflake

```
if game_over:
    display_message("GAME OVER!", "Try again.")
    t = 0
    while t < 1000000:
        t += 1
    game_over = False
    game_complete = False
    current_level = 1
    stars = []
    animations = []
```

Game Restart

```
START_SPEED = 7
COLORS = ["green",
```

Speed Change