Purpose:

#creating a ppg signal with noise.
#Filltered the ppg signal by low pass fillter.
#Detected peaks and valleys.
#Abnormal peaks detection.
And Count number of abnormal peaks,
estimated heart rate (BPM)and
total peak .

code:

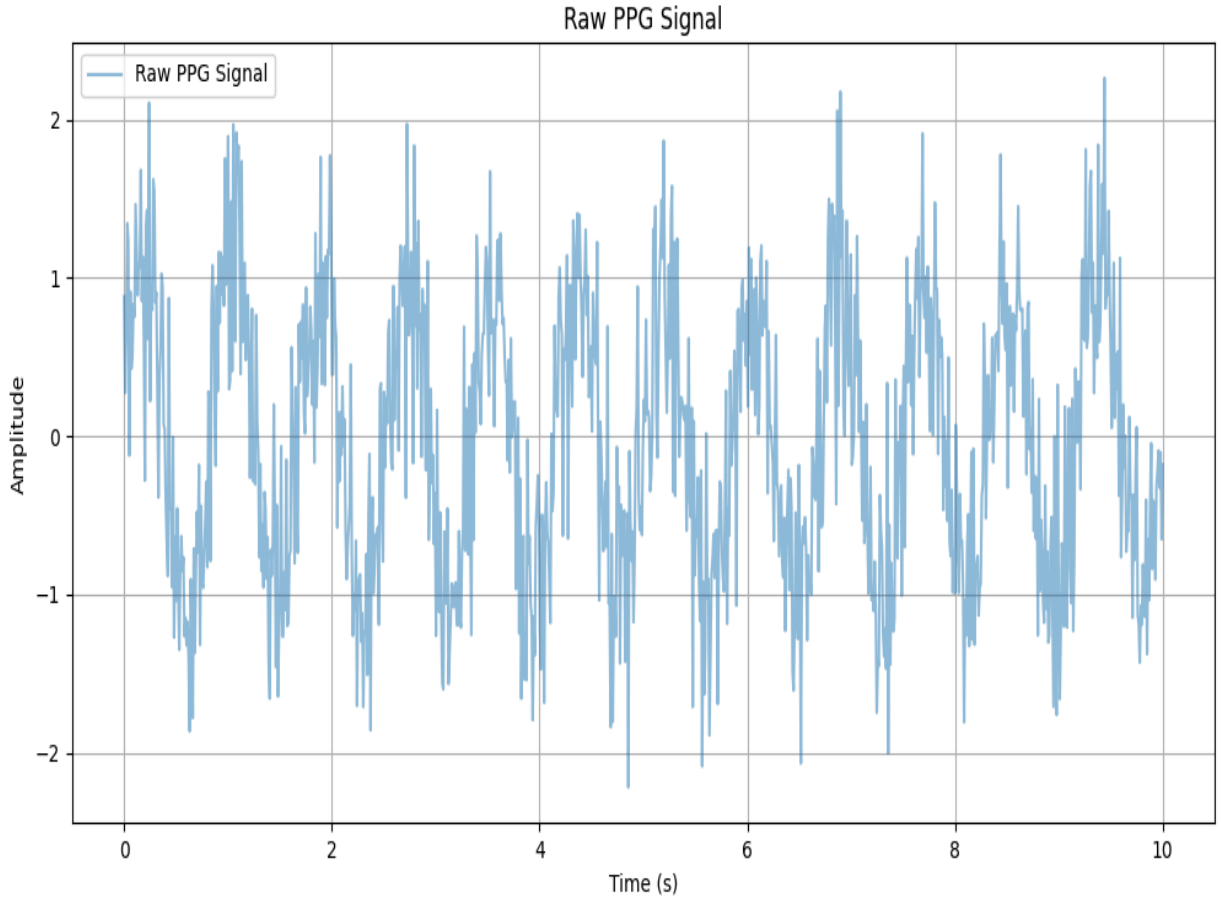
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, butter, filtfilt


# Generate sample PPG signal
np.random.seed(0)
time = np.linspace(0, 10, 1000)
ppg_signal = np.sin(2 * np.pi * 1.2 * time) + 0.5 *
np.random.normal(size=len(time))


# Define the lowpass filter function
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low',
analog=False)
    y = filtfilt(b, a, data)
    return y
```

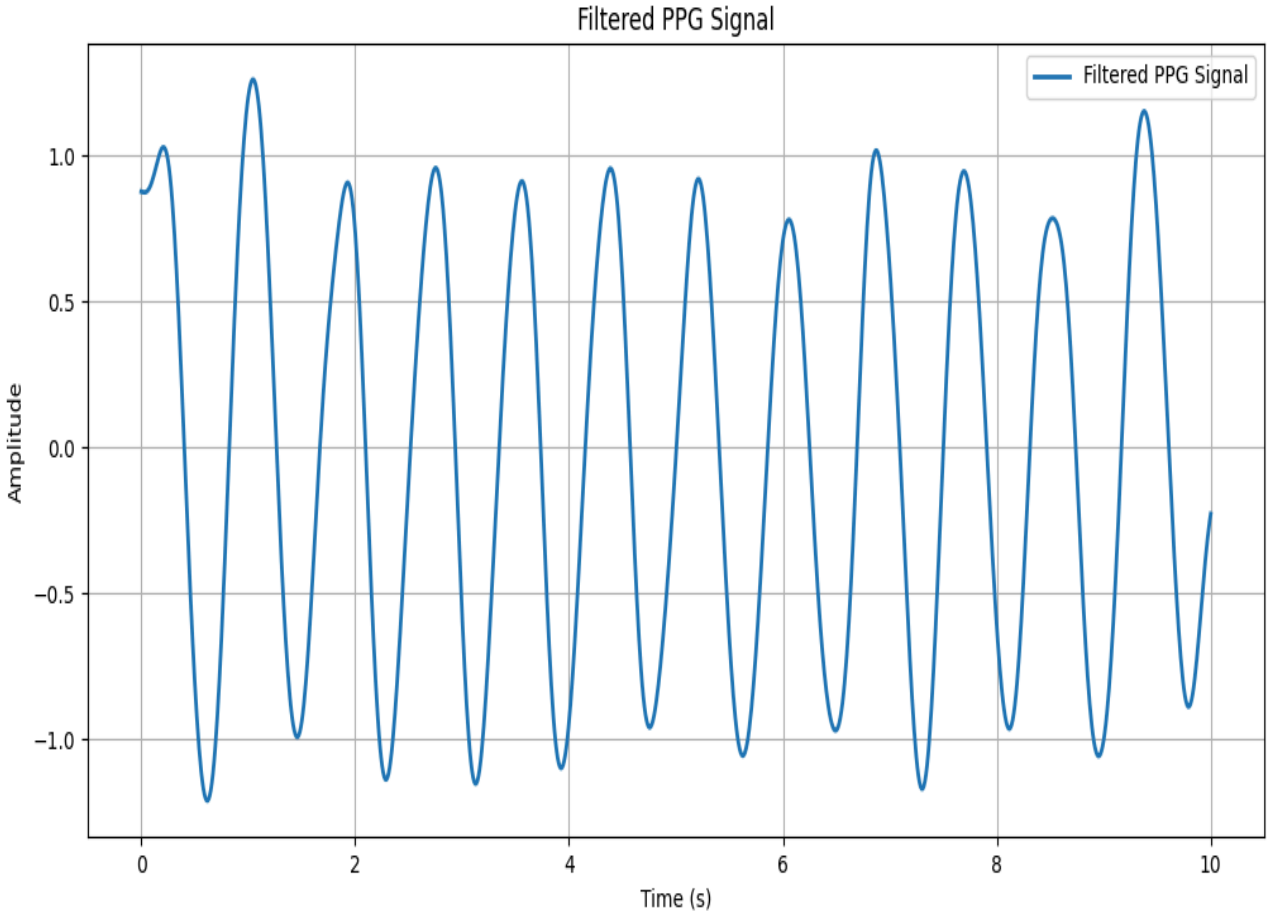
```
# Filter settings
fs = 100  # Sampling frequency in Hz
cutoff = 3  # Cutoff frequency in Hz
filtered_ppg = butter_lowpass_filter(ppg_signal, cutoff, fs)


# Peak and valley detection
peaks, _ = find_peaks(filtered_ppg, height=0.5,
distance=fs//2)
valleys, _ = find_peaks(-filtered_ppg, height=0.5,
distance=fs//2)


# Abnormal peaks detection
peak_heights = filtered_ppg[peaks]
abnormal_peaks = peaks[peak_heights > 1]  # Threshold for
high spikes


# Calculate Heart Rate (BPM)
time_diffs = np.diff(time[peaks])  # Time difference between
consecutive peaks
avg_rr_interval = np.mean(time_diffs)  # Average R-R interval
in seconds
heart_rate = 60 / avg_rr_interval  # Convert to BPM


print(f"Number of abnormal peaks: {len(abnormal_peaks)}")
print(f"Estimated Heart Rate: {heart_rate:.2f} BPM")


# Count total peak values
total_peaks = len(peaks)
print(f"Total Peak Count: {total_peaks}")


# Plot 1: Raw PPG Signal
plt.figure(figsize=(12, 6))
```

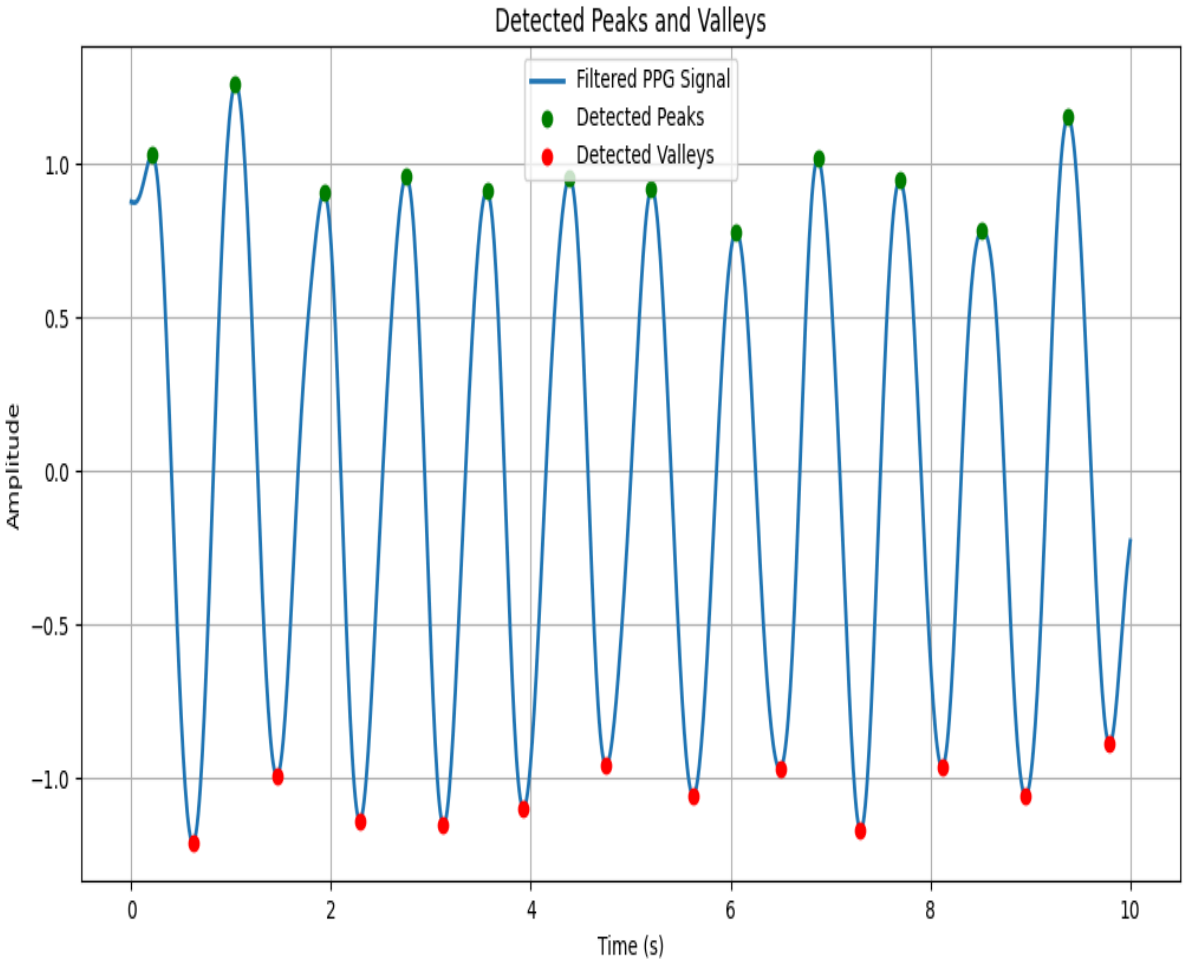
```
plt.plot(time, ppg_signal, label="Raw PPG Signal", alpha=0.5)
plt.title("Raw PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()


# Plot 2: Filtered PPG Signal
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal",
linewidth=2)
plt.title("Filtered PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()


# Plot 3: Peaks and Valleys
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal",
linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go",
label="Detected Peaks")
plt.plot(time[valleys], filtered_ppg[valleys], "ro",
label="Detected Valleys")
plt.title("Detected Peaks and Valleys")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()


# Plot 4: Abnormal Peaks
```

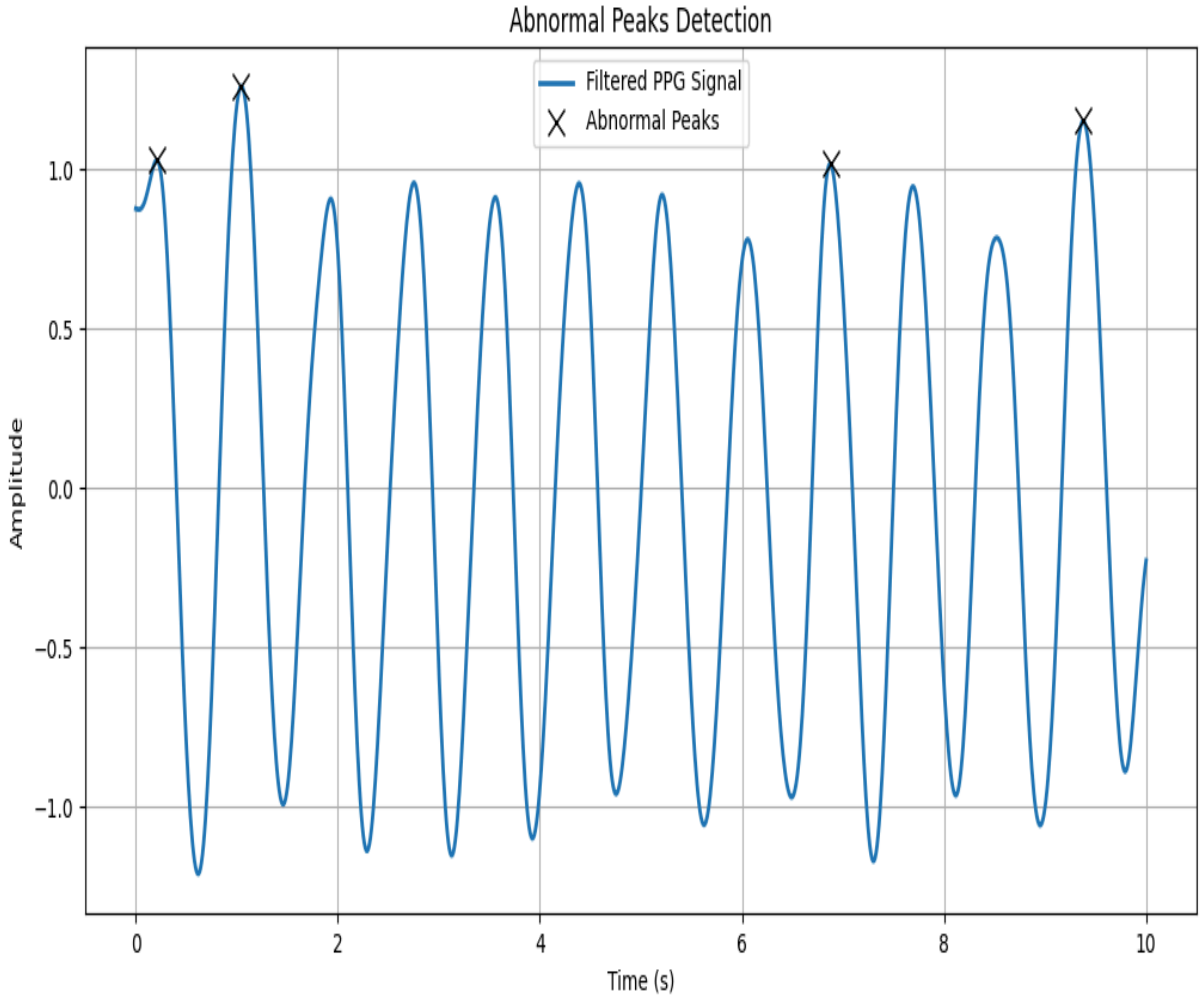
```
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal",
linewidth=2)
plt.plot(time[abnormal_peaks], filtered_ppg[abnormal_peaks],
"kx", label="Abnormal Peaks", markersize=10)
plt.title("Abnormal Peaks Detection")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

Output:


Raw PPG Signal
Raw PPG Signal
Amplitude
Time (s)
2
1
0
−1
−2
0
2
4
6
8
10


Filtered PPG Signal
Filtered PPG Signal
Amplitude
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
Time (s)


Detected Peaks and Valleys
Filtered PPG Signal
Detected Peaks
Detected Valleys
Amplitude
Time (s)
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10



Number of abnormal peaks: 4
Estimated Heart Rate: 71.98 BPM
Total Peak Count: 12