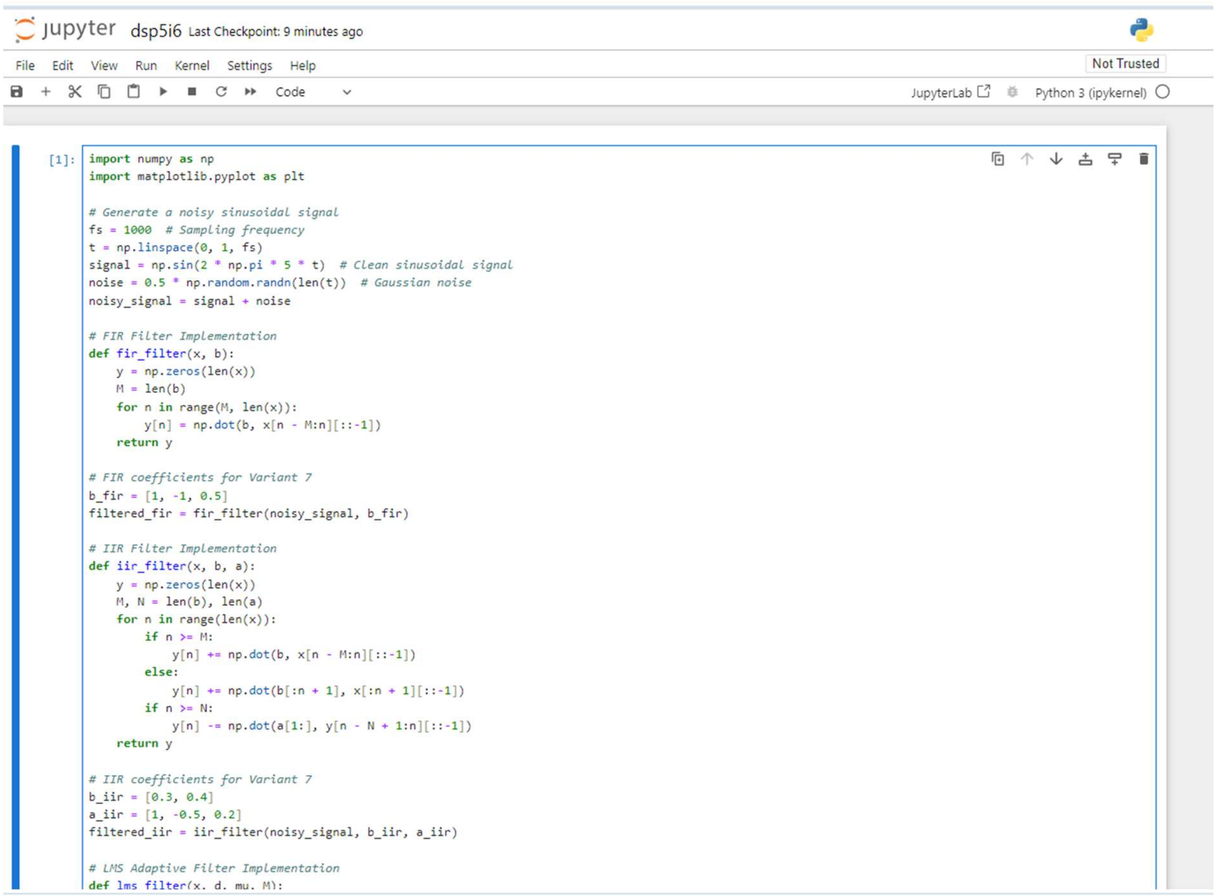
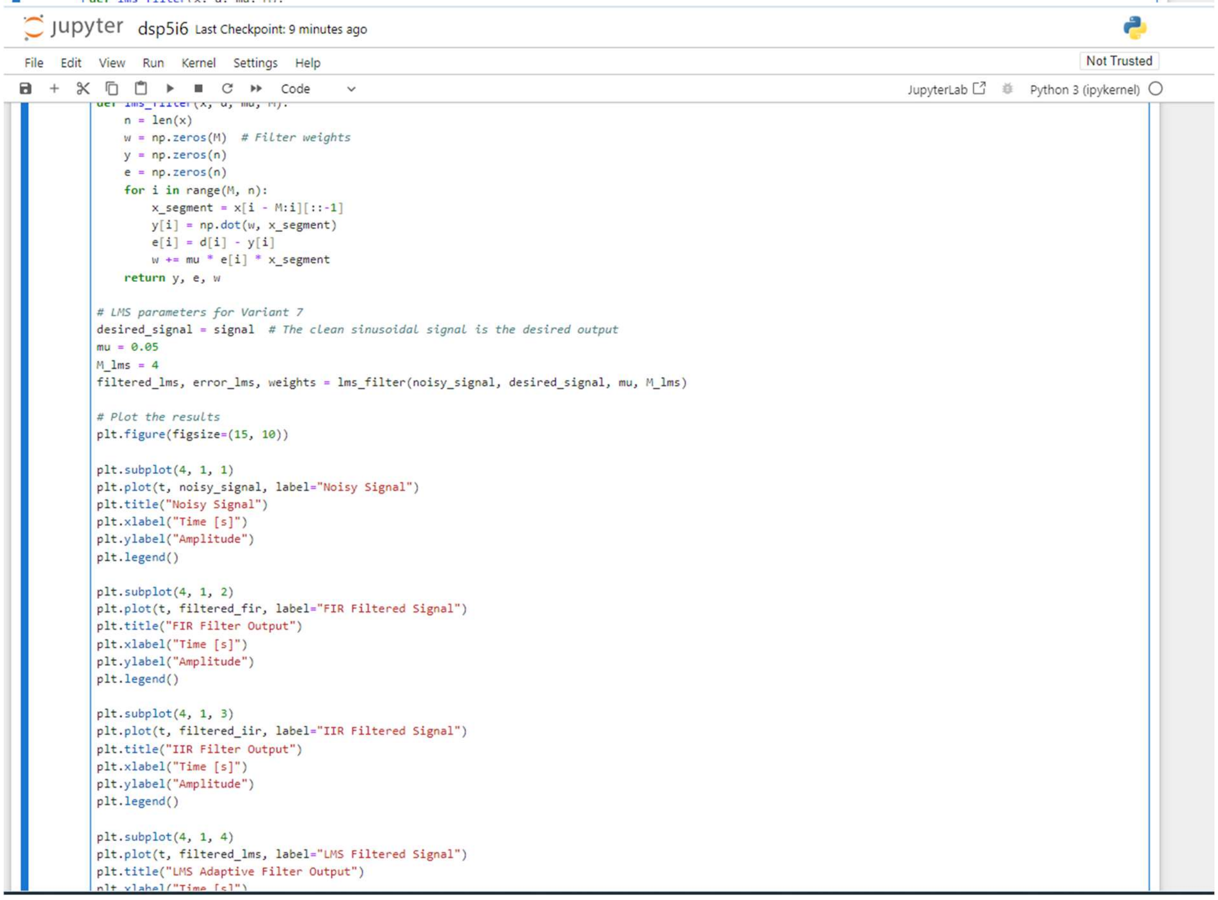
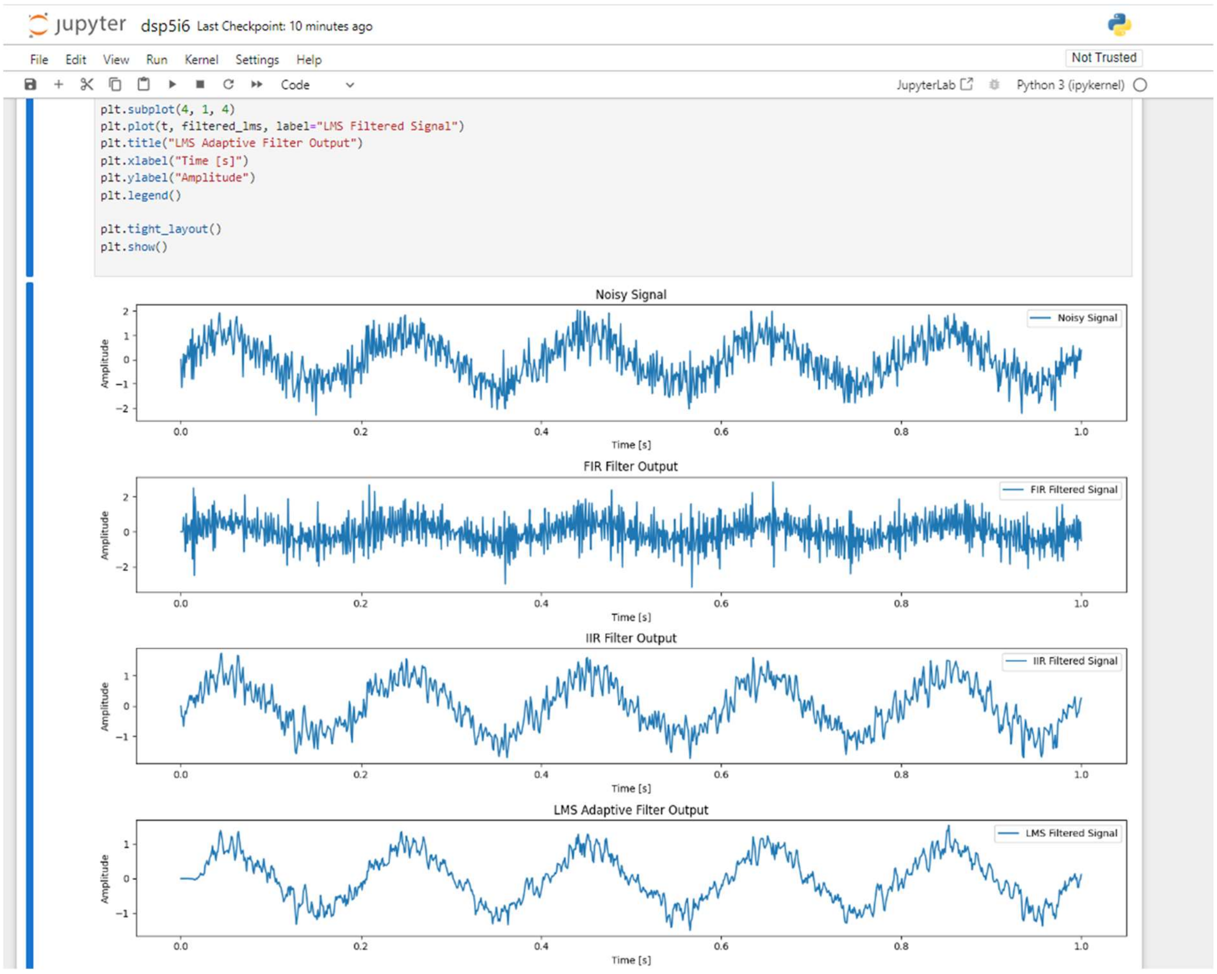

jupyter dsp5i6 Last Checkpoint: 9 minutes ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[1]: import numpy as np
     import matplotlib.pyplot as plt

     # Generate a noisy sinusoidal signal
     fs = 1000  # Sampling frequency
     t = np.linspace(0, 1, fs)
     signal = np.sin(2 * np.pi * 5 * t)  # Clean sinusoidal signal
     noise = 0.5 * np.random.randn(len(t))  # Gaussian noise
     noisy_signal = signal + noise

     # FIR Filter Implementation
     def fir_filter(x, b):
         y = np.zeros(len(x))
         M = len(b)
         for n in range(M, len(x)):
             y[n] = np.dot(b, x[n - M:n][::-1])
         return y

     # FIR coefficients for Variant 7
     b_fir = [1, -1, 0.5]
     filtered_fir = fir_filter(noisy_signal, b_fir)

     # IIR Filter Implementation
     def iir_filter(x, b, a):
         y = np.zeros(len(x))
         M, N = len(b), len(a)
         for n in range(len(x)):
             if n >= M:
                 y[n] += np.dot(b, x[n - M:n][::-1])
             else:
                 y[n] += np.dot(b[:n + 1], x[:n + 1][::-1])
             if n >= N:
                 y[n] -= np.dot(a[1:], y[n - N + 1:n][::-1])
         return y

     # IIR coefficients for Variant 7
     b_iir = [0.3, 0.4]
     a_iir = [1, -0.5, 0.2]
     filtered_iir = iir_filter(noisy_signal, b_iir, a_iir)

     # LMS Adaptive Filter Implementation
     def lms_filter(x, d, mu, M):
```


jupyter dsp5i6 Last Checkpoint: 9 minutes ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
     def lms_filter(x, d, mu, M):
         n = len(x)
         w = np.zeros(M)  # Filter weights
         y = np.zeros(n)
         e = np.zeros(n)
         for i in range(M, n):
             x_segment = x[i - M:i][::-1]
             y[i] = np.dot(w, x_segment)
             e[i] = d[i] - y[i]
             w += mu * e[i] * x_segment
         return y, e, w

     # LMS parameters for Variant 7
     desired_signal = signal  # The clean sinusoidal signal is the desired output
     mu = 0.05
     M_lms = 4
     filtered_lms, error_lms, weights = lms_filter(noisy_signal, desired_signal, mu, M_lms)

     # Plot the results
     plt.figure(figsize=(15, 10))

     plt.subplot(4, 1, 1)
     plt.plot(t, noisy_signal, label="Noisy Signal")
     plt.title("Noisy Signal")
     plt.xlabel("Time [s]")
     plt.ylabel("Amplitude")
     plt.legend()

     plt.subplot(4, 1, 2)
     plt.plot(t, filtered_fir, label="FIR Filtered Signal")
     plt.title("FIR Filter Output")
     plt.xlabel("Time [s]")
     plt.ylabel("Amplitude")
     plt.legend()

     plt.subplot(4, 1, 3)
     plt.plot(t, filtered_iir, label="IIR Filtered Signal")
     plt.title("IIR Filter Output")
     plt.xlabel("Time [s]")
     plt.ylabel("Amplitude")
     plt.legend()

     plt.subplot(4, 1, 4)
     plt.plot(t, filtered_lms, label="LMS Filtered Signal")
     plt.title("LMS Adaptive Filter Output")
     plt.xlabel("Time [s]")
```


jupyter dsp5i6 Last Checkpoint: 10 minutes ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
plt.subplot(4, 1, 4)
plt.plot(t, filtered_lms, label="LMS Filtered Signal")
plt.title("LMS Adaptive Filter Output")
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()

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