```python
# Author: Ramesh Pai
# Affiliation: 201104047, TE-E&TC Engg, Sem.5, 2021-22, GCE.

# importing python modules
import matplotlib.pyplot as plt
import numpy as np
from scipy.fft import fft, fftshift, ifftshift, ifft

# vm1: Amplitude of Message Signal 1
# vm2: Amplitude of Message Signal 2
# fm1: fequency(Hz) of Message Signal 1
# fm2: fequency(Hz) of Message Signal 2

def values(vm1, vm2, fm1, fm2):

    fs = 60000 #sampling frequency
    dt = 1/fs #sample time interval or time-steps for time-domain signal
    t = np.arange(0, 0.2, dt) #time indices for time-domain signal
    n = np.size(t) #number of samples
    df = fs/n #frquency interval or frequency-steps for frequency-spectrum
    f = np.arange(-fs/2, fs/2, df) #frequency indices for frquency-spectrum

    # plot1: Message Signal 1/Sinusoid(Volts) v/s Time(sec)
    v1 = vm1*np.cos(2*np.pi*fm1*t)
    plt.subplot(3, 3, 1)
    plt.plot(t, v1)
    plt.title("Message Signal 1", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v1(Volts)")

    # plot2: Message Signal 2/Sinusoid(Volts) v/s Time(sec)
    v2 = vm2*np.cos(2*np.pi*fm2*t)
    plt.subplot(3, 3, 2)
    plt.plot(t, v2)
    plt.title("Message Signal 2", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v2(Volts)")

    # plot3: Spectrum of Message Signal 1(Magnitude) v/s Frequency(Hz)
    xf1 = fftshift(fft(v1)) #FFT of Message Signal 1(Complex in nature)
    plt.subplot(3, 3, 3)
    plt.plot(f, abs(xf1)/n) #PLotting frequency indices v/s Normalised magnitude of FFT Message signal 1
    plt.xlim(400, -400)
    plt.title("Message 1 frequency Spectrum", loc='left')
    plt.xlabel("frequency(Hz)", loc='right')
    plt.ylabel("Magnitude")

    # plot4: Spectrum off Message Signal 2(Magnitude) v/s Frequency(Hz)
    xf2 = fftshift(fft(v2)) #FFT of Message Signal 2(Complex in nature)
    plt.subplot(3, 3, 4)
    plt.plot(f, abs(xf2)/n) #PLotting frequency indices v/s Normalised magnitude of FFT Message signal 2
    plt.xlim(400, -400)
    plt.title("Message 2 frequency Spectrum", loc='left')
    plt.xlabel("frequency(Hz)", loc='right')
    plt.ylabel("Magnitude")

    # plot5: Spectrum of FDM Signal v/s Frequency(Hz)
```

```python
    xf3 = xf1 + xf2 #Frequency Division Multiplexing both Message Signals
    plt.subplot(3, 3, 5)
    plt.plot(f, abs(xf3)/n)  #PLotting frequency indices v/s Normalised magnitude of FDM Signal
    plt.xlim(400, -400)
    plt.title("Spectrum of FDM Signal", loc='left')
    plt.xlabel("frequency(Hz)", loc='right')
    plt.ylabel("Magnitude")

    # plot6: Demultiplexed signals v/s Time(sec)

    # filter 1 Designing
    bpf1 = [] #List having array of 0's and 1's

    for x in f:
        if x < fm1+5 and x > -fm1-5:
            x = 1
            bpf1.append(x) #Assigning 1 to frequencies in the Range/Band
        else:
            x = 0
            bpf1.append(x) #Assigning 0 to frequencies not the Range/Band

    #Demultiplexing to get Message Signal 1
    y1 = xf3*bpf1 #Multiplying Filter 1 with FDM Signal to Aquire Original Message 1 Signal
    dm1 = ifft(ifftshift(y1)) #Inverse FFT to get Original Message signal 1(time-domain)
    plt.subplot(3, 3, 6)
    plt.plot(t, dm1)
    plt.title("Demultiplexed Message Signal 1", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v1(Volts)")

    # filter 2 Designing
    bpf2 = [] #List having array of 0's and 1's

    for x in f:
        if x  > -(fm2+5) and x < -(fm1+5) or x < (fm2+5) and x > (fm1+5):
            x = 1
            bpf2.append(x) #Assigning 1 to frequencies in the Range/Band
        else:
            x = 0
            bpf2.append(x) #Assigning 0 to frequencies not the Range/Band

    # Demultiplexing to get Message Signal 2
    y2 = xf3*bpf2 #Multiplying Filter 2 with FDM Signal to Aquire Original Message 2 Signal
    dm2 = ifft(ifftshift(y2)) #Inverse FFT to get Original Message signal 2(time-domain)
    plt.subplot(3, 3 , 7)
    plt.plot(t, dm2)
    plt.title("Demultiplexed Message Signal 2", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v2(Volts)")

    plt.subplot_tool()
    plt.show()

values(1, 1, 100, 150) #Assigning Values to the parameters
```