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

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

# vc: Amplitude of Carrier Signal
# fc: frequency(Hz) of Carrier Signal
# Td: Number of bits in the Binary Data(Unipolar NRZ)

def values(vc, fc, Td):

    fs = 600 #sampling frequency
    dt = 1/fs #sample time interval or time-steps for time-domain signal
    t = np.arange(0, 5, 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: Carrier Signal v/s Time

    v_c = vc*np.sin(2*np.pi*fc*t) #Carrier Signal
    plt.subplot(2, 2, 1)
    plt.plot(t, v_c)
    plt.title("Carrier Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v_c(Volts)")

    # plot2: Modulating Signal(Unipolar NRZ) v/s Time

    #list for storing random input bit sequence generated
    bitData = []
    for x in range(Td):
        x = random.randint(0, 1)
        bitData.append(x)

    print(f"input bit sequence: {bitData}") #printing the input bit sequence

    #number of points in the time duration
    Td_len = int(len(t)/Td)

    #list for storing number of points in time duration
    Td_arr = []
    for x in range(Td+1):
        x *= Td_len
        Td_arr.append(x)

    sqSignal = [] #Modulating Signal(Unipolar NRZ)
    for x in range(len(bitData)):
        if bitData[x] == 1:
            x = [1]*(Td_arr[x+1] - Td_arr[x])
            sqSignal.extend(x)
        else:
            x = [0]*(Td_arr[x+1] - Td_arr[x])
            sqSignal.extend(x)

```

```
    sqSignal = np.array(sqSignal) #converting list into numpy array
    plt.subplot(2, 2, 2)
    plt.plot(t, sqSignal)
    plt.title(f"Modulating Signal (Binary Data: {bitData})", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("sqSignal(Volts)")

    # plot3: ASK(Amplitude Shift Keying) Signal v/s Time

    v_a = v_c * sqSignal # ASK(Amplitude Shift Keying) Signal
    plt.subplot(2, 2, 3)
    plt.plot(t, sqSignal, color = 'r', linestyle = 'dotted', label = 'modulating signal')
    plt.plot(t, v_a, label = 'ASK Signal')
    plt.title("ASK Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v_a(Volts)")

    # plot4: Spectrum of the ASK(Amplitude Shift Keying) Signal v/s Frequency

    v_a_spec = fftshift(fft(v_a)) #FFT of ASK Signal(Complex in nature)
    plt.subplot(2, 2, 4)
    plt.plot(f, abs(v_a_spec)/n) #PLotting frequency indices v/s Normalised magnitude of FFT ASK signal
    plt.xlim(-20, 20)
    plt.title("Spectrum of ASK Signal", loc='left')
    plt.xlabel("frequency(Hz)", loc='right')
    plt.ylabel("Magnitude")

    plt.subplot_tool()
    plt.show()

values(1, 3, 5) #assigning values to parameters


```