**ST.FRANCIS INSTITUTE OF TECHNOLOGY**

Mount Poinsur, S.V.P. Road, Borivli (West), Mumbai - 400103

**Computer Engineering Department**

**Academic Year:** 2021-2022 **Class/Branch**: BE COMP

**Subject:** CSL 701 Digital Signal and Image Processing **Semester**: VII

Experiment No 03: Basic Operations on Signals

CLASS: BE CMPN A ROLL NO. : 19

Name: Rebecca Dias Pid: 182027

**Aim :** To perform the basic signal operations in Python

Theory :

# Basic Operations on Signals

The discrete time sequence may undergo several manipulations involving the independent variable or the amplitude of the signal.

The basic operations on sequences are as follows:

1. Time shifting
2. Time reversal
3. Time scaling
4. Amplitude scaling
5. Signal addition
6. Signal multiplication

The first three operations correspond to transformation in independent variable n of a signal. The last three operations correspond to transformation on amplitude of a signal.

### Time Shifting

The time shifting of a signal may result in time delay or time advance. The time shifting operation of a discrete-time signal x(n) can be represented by



This shows that the signal y(n) can be obtained by time shifting the signal x(n) by k units. If k is positive, it is delay and the shift is to the right, and if k is negative, it is advance and the shift is to the left.

### Time Reversal

The time reversal also called time folding of a discrete-time signal x(n) can be obtained by folding the sequence about n = 0. The time reversed signal is the reflection of the original signal. It is obtained by replacing the independent variable n by –n.

### Time Scaling

Time scaling may be time expansion or time compression. The time scaling of a discrete time signal x(n) can be accomplished by replacing **n** by **an** in it. Mathematically, it can be expressed as:



When a > 1, it is time compression and when a < 1, it is time expansion. Time scaling is very useful when data is to be fed at some rate and is to be taken out at a different rate.

### Amplitude Scaling (Scalar Multiplication)

Amplitude scaling of a signal by a constant A is obtained by multiplying the value of every signal sample by A.

y(n) = A x(n)

The amplitude of y(n) at any instant is equal to a times the amplitude of x(n) at that

instant. If a > 1, it is amplification and if a < 1, it is attenuation. Hence the amplitude is rescaled. Hence the name amplitude scaling.

### Signal Addition

In discrete-time domain, the sum of two signals x1(n) and x2(n) can be obtained by adding the corresponding sample values.

### Signal Multiplication

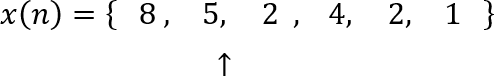
In a discrete-time domain, the product of two signals x1(n) and x2(n) can be obtained by multiplying the corresponding sample values.

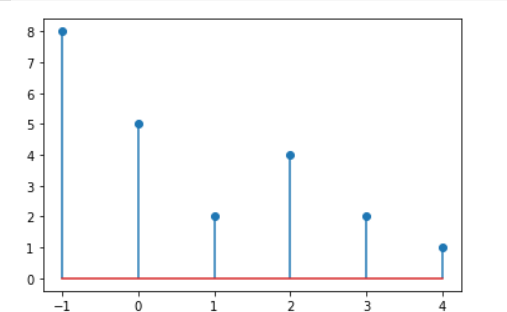


**For the signal given below, plot the following signals**

(i) x(n+2) (ii)x(n)u(-n) (iii)x(n-1)u(-n-2)

(iv) x(-n-1)u(n) (v) x(2n-1)



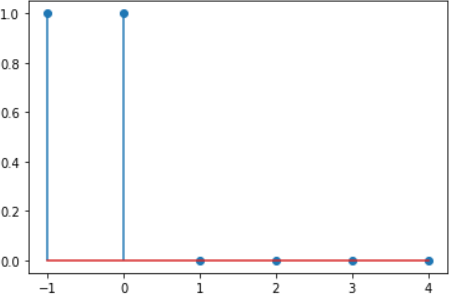
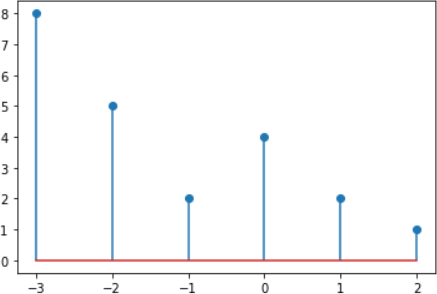


import numpy as np

import matplotlib.pyplot as plt x=[8,5,2,4,2,1]

n=np.linspace(-1,4,6) plt.stem(n,x,use\_line\_collection=True);

plt.stem(n-2,x,use\_line\_collection=True);

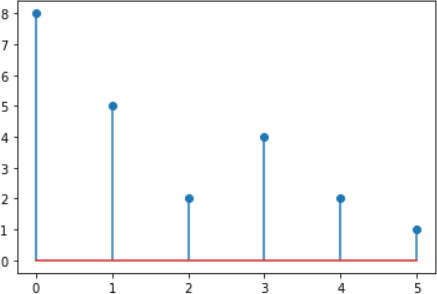
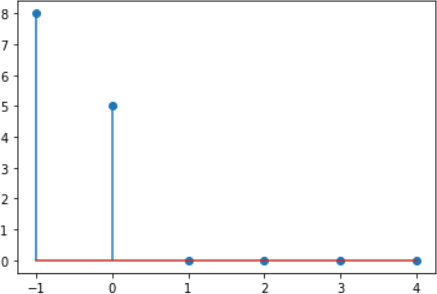


x1 = np.linspace(-1,4,6) y1 = (x1<=0)\*1

plt.stem(x1,y1,use\_line\_collection=True) plt.show()

x2 = np.linspace(-1,4,6) y2=x\*y1

plt.stem(x2,y2,use\_line\_collection=True) plt.show()



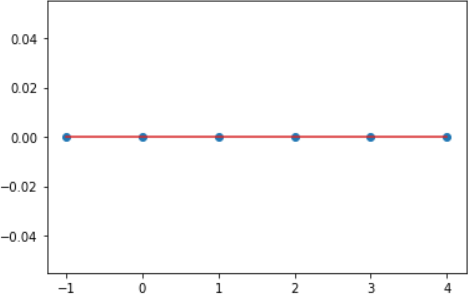
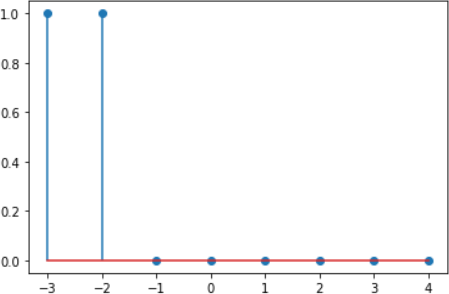
x=[8,5,2,4,2,1]

n=np.linspace(-1,4,6) n1=n+1

plt.stem(n1,x,use\_line\_collection=True);

x2 = np.linspace(-3,4,8) y2 = (x2<=-2)\*1

plt.stem(x2,y2,use\_line\_collection=True) plt.show()



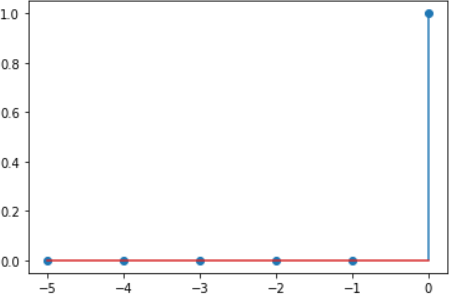
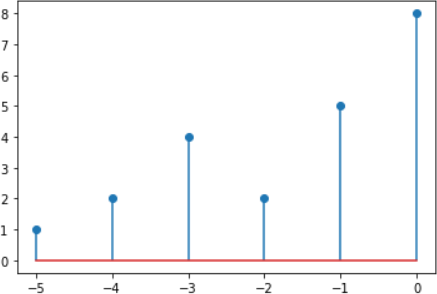
x1=np.linspace(-1,4,6) y1 = (x1<=-2)\*1

plt.stem(x1,y1,use\_line\_collection=True) plt.show()

x=[8,5,2,4,2,1]

n=np.linspace(-1,4,6) n2=-n-1

plt.stem(n2,x,use\_line\_collection=True);



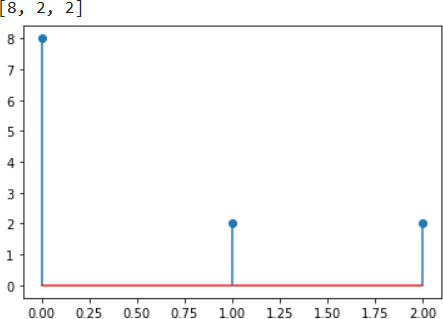
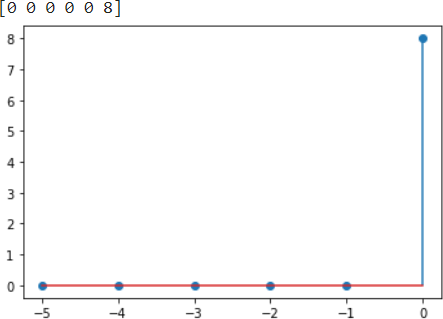
#u(n)

x1 = np.linspace(-5,0,6) y1 = (x1>=0)\*1

plt.stem(x1,y1,use\_line\_collection=True) plt.show()

y2=y1\*x[0] print(y2)

plt.stem(x1,y2,use\_line\_collection=True) plt.show()



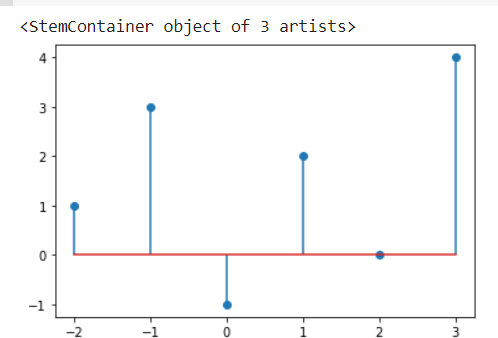
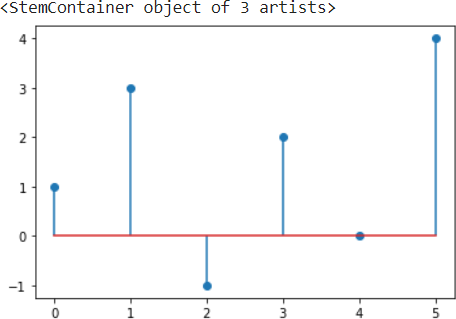
x=[8,5,2,4,2,1]

res = [x[i] for i in range(len(x)) if i % 2-1!= 0] print(res)

n=np.linspace(0,2,3) plt.stem(n,res,use\_line\_collection=True);

# For x(n) = { 1,3,-1,2,0,4 }, plot the following signals

## (i) x(n+2) (ii)x(-n-1) (iii)2x(n)



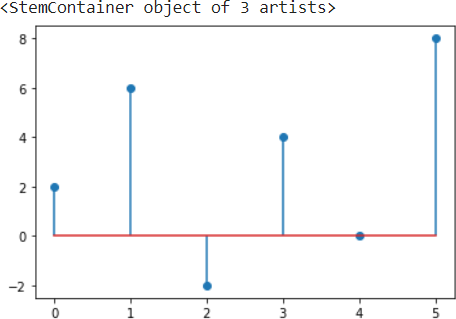
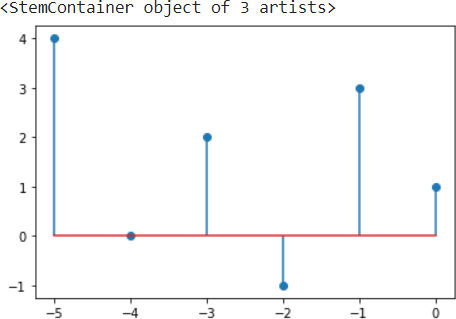
(iv) x(n-1)𝛿(n-3) (v) x(n)u(n-2)

x = [1,3,-1,2,0,4]

n = np.linspace(0,5,6) plt.stem(n,x,use\_line\_collection=True)

plt.stem(n-2,x,use\_line\_collection=True)

x = [1,3,-1,2,0,4]



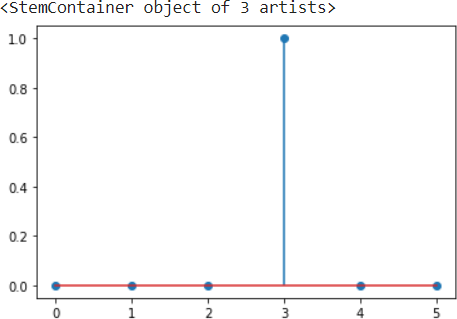
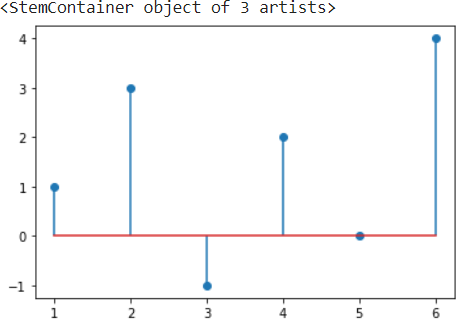
n = np.linspace(-1,4,6) n2 = -n-1

plt.stem(n2,x,use\_line\_collection=True)

x = [2\*i for i in x] n=np.linspace(0,5,6) plt.stem(n,x,use\_line\_collection=True)

x = [1,3,-1,2,0,4]

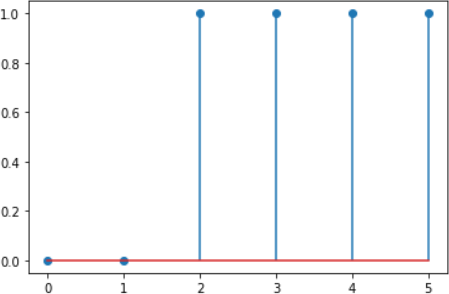
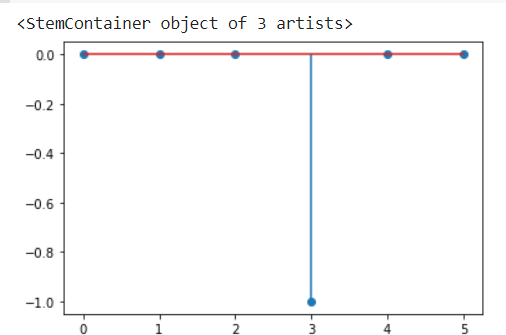
n = np.linspace(0,5,6) n1 = n+1



plt.stem(n1,x,use\_line\_collection=True)

x1 = [1 if i==3 else 0 for i in range(len(x))] n = np.linspace(0,5,6) plt.stem(n,x1,use\_line\_collection=True)

x2 = [x[i-1] if x1[i]==1 else 0 for i in range(len(x1))] n = np.linspace(0,5,6) plt.stem(n,x2,use\_line\_collection=True)

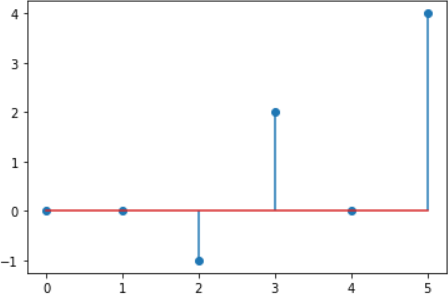


x1 = np.linspace(0,5,6) y1 = (x1>=2)\*1

plt.stem(x1,y1,use\_line\_collection=True) plt.show()

x2 = np.linspace(0,5,6) y2 = x\*y1

plt.stem(x2,y2,use\_line\_collection=True) plt.show()



**Conclusion:** We learnt that Time shifting, reversal and scaling operations correspond to transformation in independent variable n of a signal while amplitude scaling, signal addition and signal multiplication operations correspond to transformation on amplitude of a signal. We implemented these basic operations on discrete time signals using Python.