DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Digital Signal Processing

ENCS4310

Assignment (1)

Student Name: Mahmoud Qaisi

Student ID: 1190831

Instructor: Dr. Qadri Mayyala.

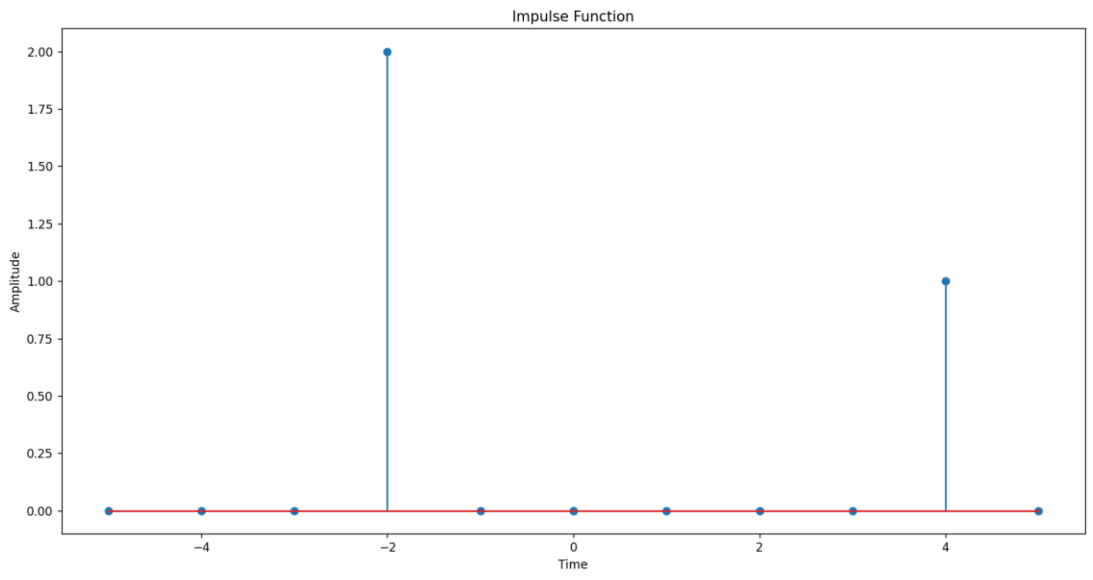
Section: 2

Date: 13/1/2023

- Question 1:

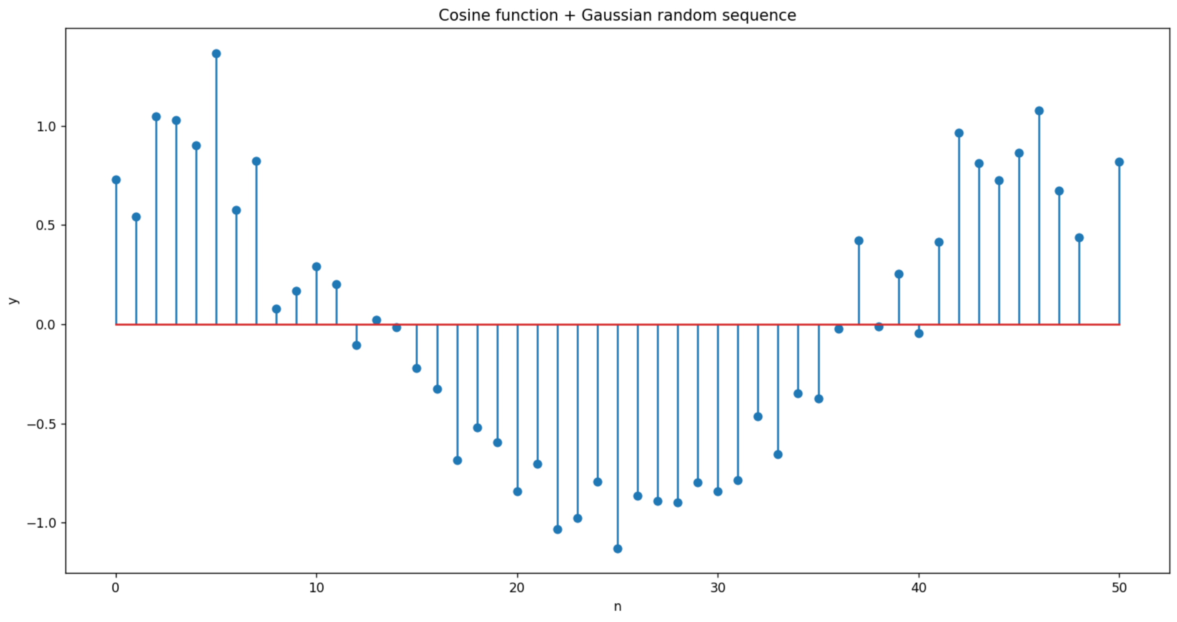
1- $x[n] = 2\delta(n+2) - \delta(n-4), \ -5 \le n \le 5.$

```
import matplotlib.pyplot as plt
import numpy as np


# Define the impulse function
def impulse(t, d):
    return 1 if t - d == 0 else 0


n = np.linspace(-5, 5, 12).astype(int)

y = [2 * impulse(i, -2) + impulse(i, 4) for i in n]

# Plot the impulse function
plt.stem(n, y)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Impulse Function')
plt.show()
```



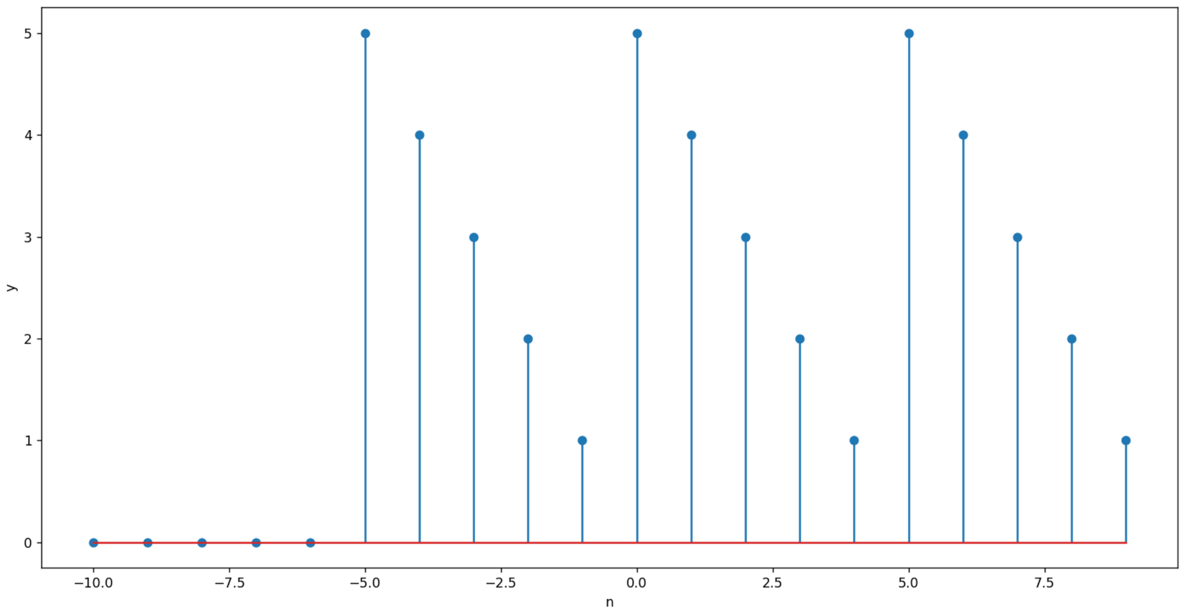
2- $y[n] = \cos(0.04\pi n) + 0.2w(n),\ 0 \leq n \leq 50.$

```
import numpy as np
import matplotlib.pyplot as plt

n = np.linspace(0, 50, 50).astype(int)

y = np.cos(0.04*np.pi*n) + 0.2 * (np.random.randn(50))

plt.stem(n, y)
plt.xlabel('n')
plt.ylabel('y')
plt.title('Cosine function + Gaussian random sequence')
plt.show()
```



3- $z[n] = \{\ldots,\ 5,4,3,2,1,\underline{5},4,3,2,1,5,4,3,2,1,\ \ldots\};\ -10 \le n \le 9,$

```
import numpy as np
import matplotlib.pyplot as plt

n = np.linspace(-10, 9, 20).astype(int)
z = np.array([5,4,3,2,1,5,4,3,2,1,5,4,3,2,1])
z = np.hstack([np.zeros((n.shape[0] - z.shape[0], )), z])

plt.stem(n, z)
plt.xlabel('n')
plt.ylabel('y')
plt.show()
```

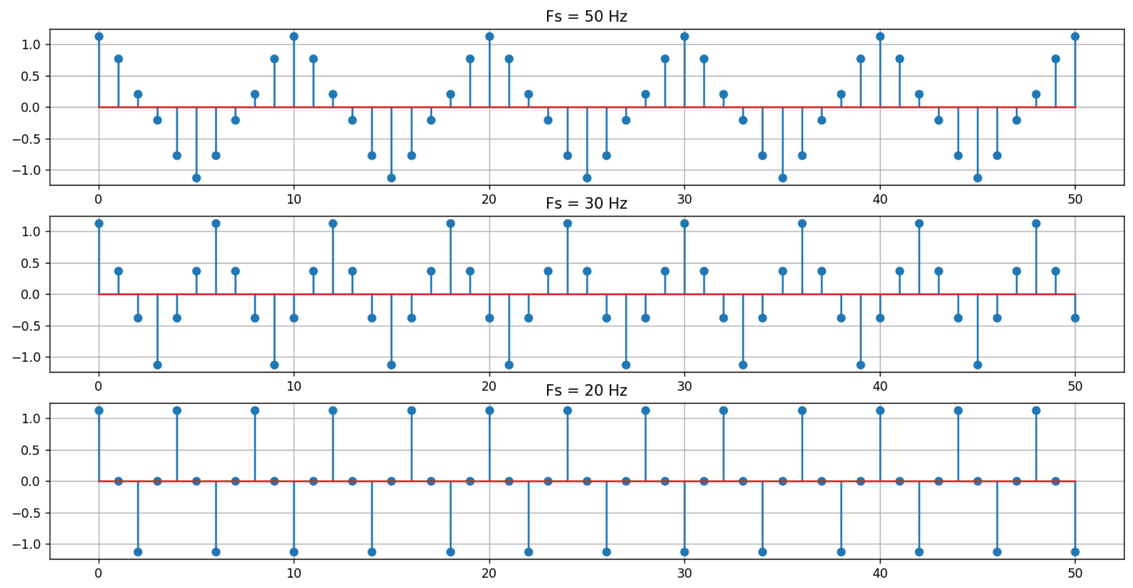


- ## Question 2:

$g(t) = cos\left(2\pi F_1 t\right) + 0.125cos\left(2\pi F_2 t\right), F_1 = 5Hz, F_2 = 15,$, plot g[n] for one second.

A) For $Fs = 50Hz$
B) For $Fs = 30Hz$
C) For $Fs = 20Hz$

```
import matplotlib.pyplot as plt
import numpy as np

n = np.linspace(0, 50, 51)
g1 = np.cos(2*np.pi*5*n/50) + 0.125*np.cos(2*np.pi*15*n/50)
g2 = np.cos(2*np.pi*5*n/30) + 0.125*np.cos(2*np.pi*15*n/30)
g3 = np.cos(2*np.pi*5*n/20) + 0.125*np.cos(2*np.pi*15*n/20)

fig, axs = plt.subplots(3, 1)

axs[0].stem(n, g1)
axs[0].grid(True)
axs[0].set_title('Fs = 50 Hz')

axs[1].stem(n, g2)
axs[1].grid(True)
axs[1].set_title('Fs = 30 Hz')

axs[2].stem(n, g3)
axs[2].grid(True)
axs[2].set_title('Fs = 20 Hz')

plt.show()
```

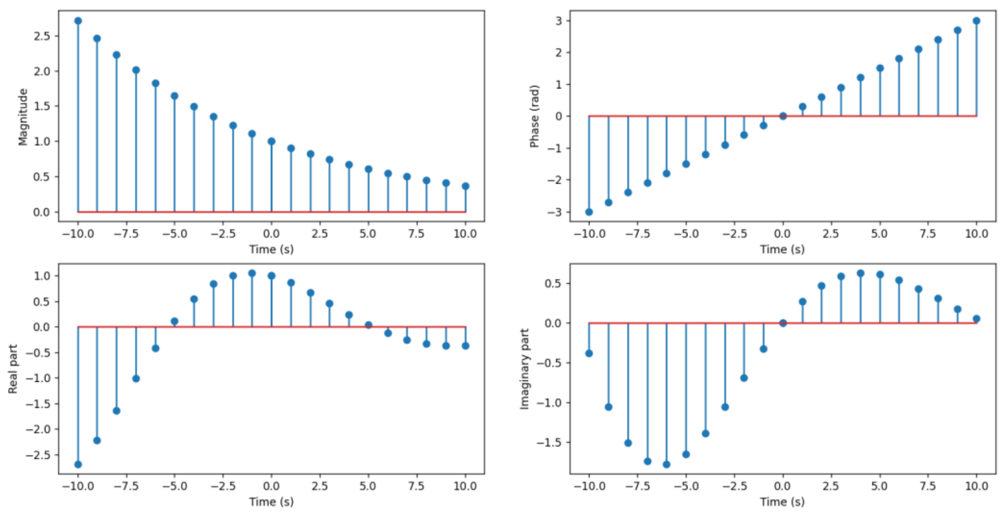


The different sampling rate affects the discrete samples that are taken from the signal. When the sampling rate is higher, more samples are taken per second and the signal appears more accurate and smoother. When the sampling rate is lower, fewer samples are taken per second and the signal appears more discrete and less accurate.

- Question 3:

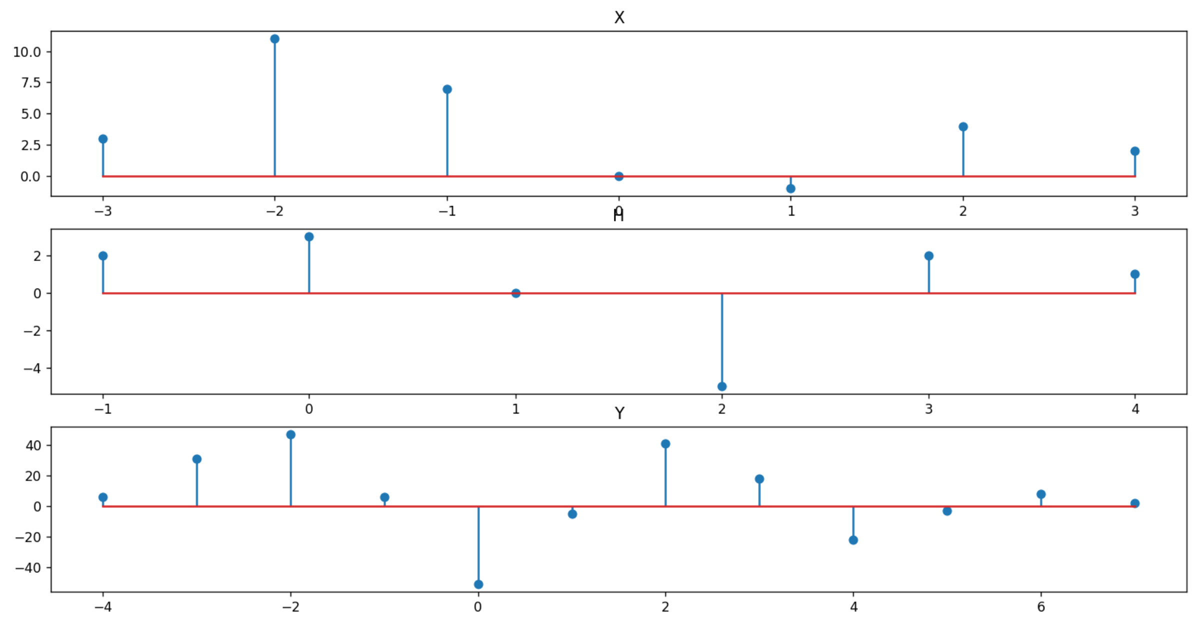
$$x[n] = e^{(-0.1+j\,0.3)n}, \quad -10 \le n \le 10$$

```
import numpy as np
import matplotlib.pyplot as plt

n = np.linspace(-10, 10, 22).astype(int)
x = np.exp((0.3j - 0.1)*n)

plt.subplot(221)
plt.stem(n, np.abs(x))
plt.xlabel('Time (s)')
plt.ylabel('Magnitude')

plt.subplot(222)
plt.stem(n, np.angle(x))
plt.xlabel('Time (s)')
plt.ylabel('Phase (rad)')

plt.subplot(223)
plt.stem(n, np.real(x))
plt.xlabel('Time (s)')
plt.ylabel('Real part')

plt.subplot(224)
plt.stem(n, np.imag(x))
plt.xlabel('Time (s)')
plt.ylabel('Imaginary part')

plt.show()
```



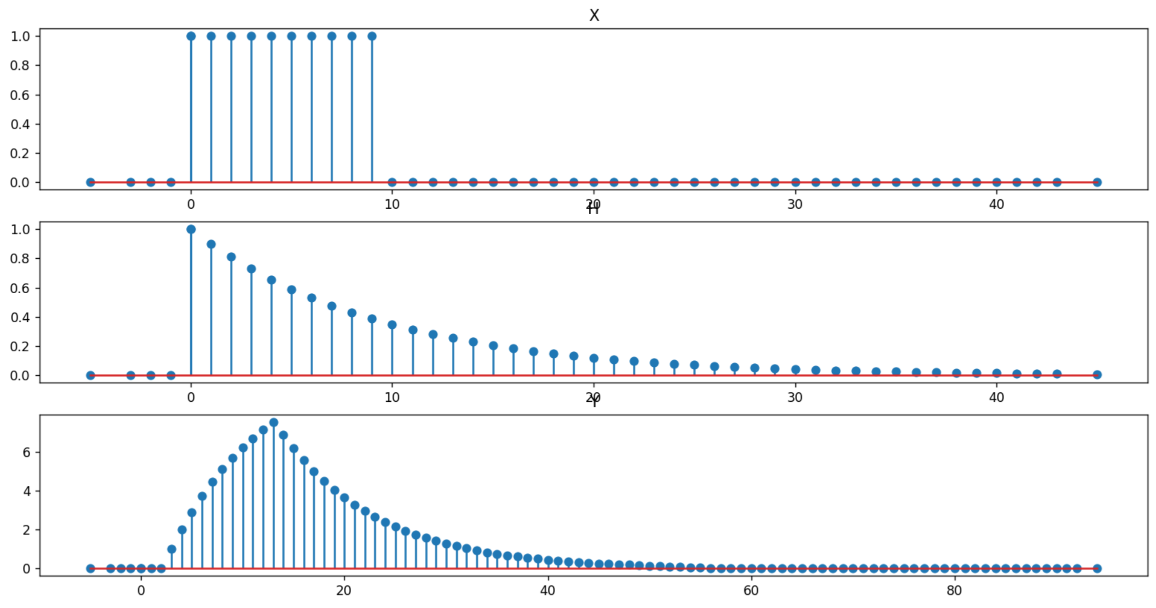
- Question 5:

```
import numpy as np
import matplotlib.pyplot as plt

nx = np.linspace(-3, 3, 7).astype(int)
x = np.array([3, 11, 7, 0 , -1, 4, 2])

nh = np.linspace(-1, 4, 6).astype(int)
h = np.array([2, 3, 0, -5, 2, 1])

ny = np.linspace(nx[0] + nh[0], nx[len(nx)-1] + nh[len(nh)-1]).astype(int)
ny = np.unique(ny)
y = np.convolve(x,h)
figure, axis = plt.subplots(3, 1)

axis[0].stem(nx, x)
axis[0].set_title("X")

axis[1].stem(nh, h)
axis[1].set_title("H")

axis[2].stem(ny, y)
axis[2].set_title("Y")

plt.show()
```



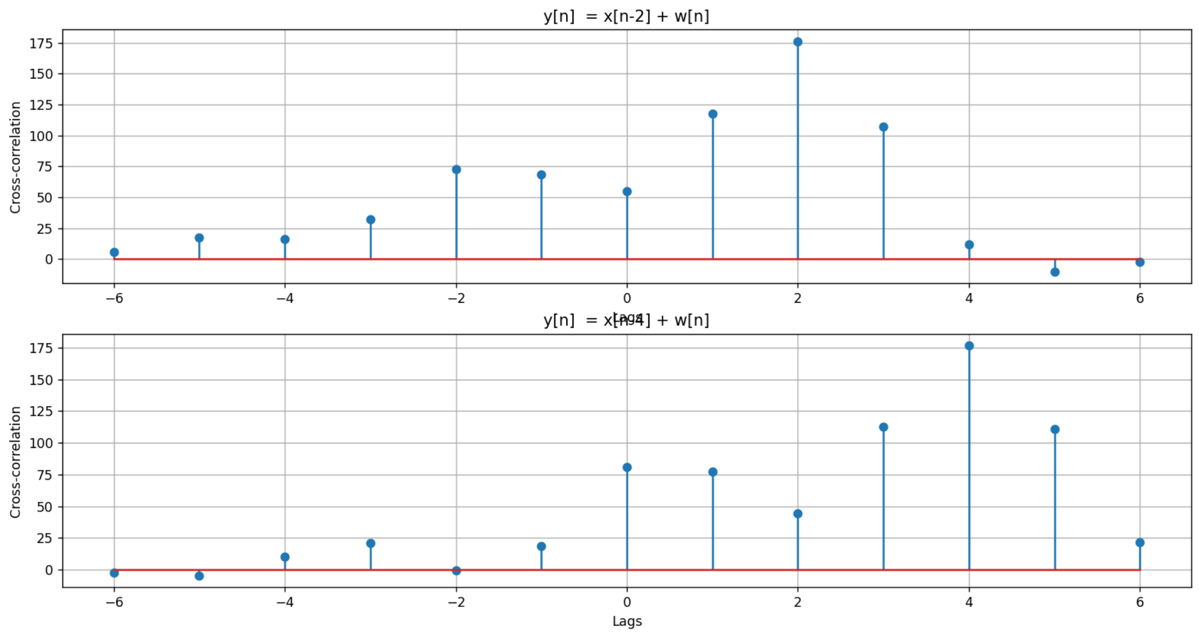
- Question 6:

```
import numpy as np
import matplotlib.pyplot as plt

n = np.linspace(-5,45).astype(int)
x = np.zeros(n.shape)
x[np.argwhere(n == 0)[0][0]:np.argwhere(n == 10)[0][0]] = 1

h = np.zeros(n.shape)
h[np.argwhere(n == 0)[0][0]:] = 1
h = h* 0.9**n

y = np.convolve(x,h)
n2 = np.linspace(-5,45+len(h)-1,len(y)).astype(int)
figure, axis = plt.subplots(3, 1)

axis[0].stem(n, x)
axis[0].set_title("X")

axis[1].stem(n, h)
axis[1].set_title("H")

axis[2].stem(n2, y)
axis[2].set_title("Y")

plt.show()
```



- Question 7:

```python
import numpy as np
import matplotlib.pyplot as plt

x = [3, 11, 7, 0, -1, 4, 2]
w = np.random.randn(len(x))
print(w)
y = [x[i - 2] + w[i] for i in range(len(x))]

r = np.correlate(y, x, mode='full')
lags = np.arange(len(r)) - len(x) + 1

y2 = [x[i - 4] + w[i] for i in range(len(x))]
r2 = np.correlate(y2, x, mode='full')
lags2 = np.arange(len(r)) - len(x) + 1

plt.subplot(211)
plt.stem(lags, r)
plt.grid()
plt.xlabel('Lags')
plt.ylabel('Cross-correlation')

plt.subplot(212)
plt.stem(lags2, r2)
plt.grid()
plt.xlabel('Lags')
plt.ylabel('Cross-correlation')
plt.show()
```



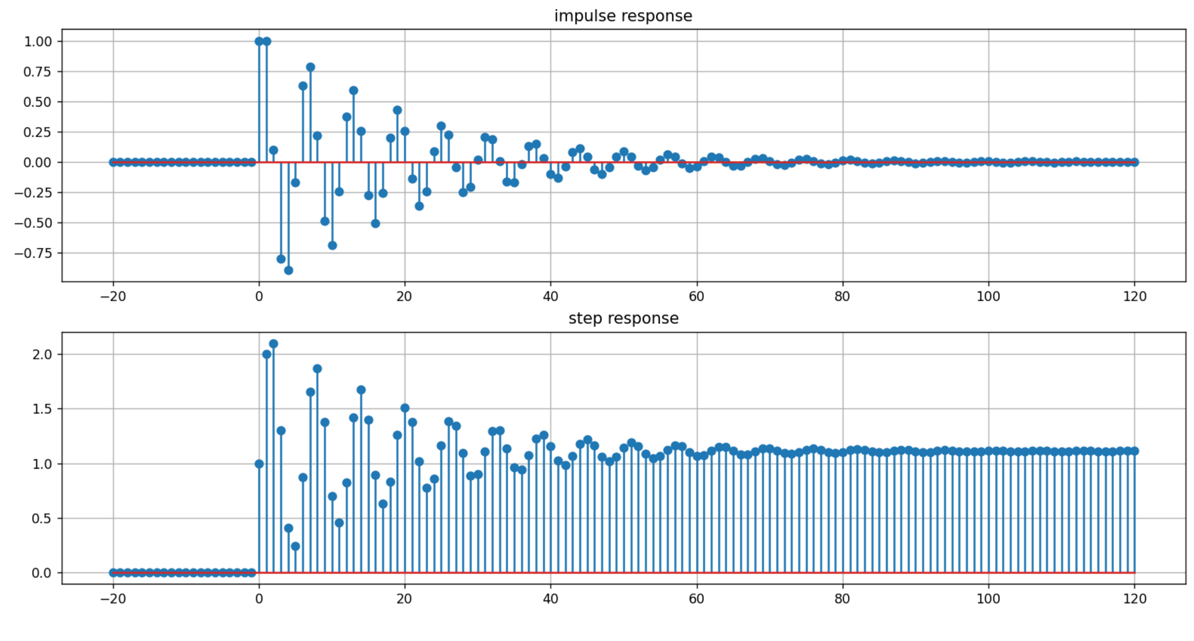
The cross-correlation values in the first case are high when the lags are 2, 1, 3 and 0, indicating that y[n] and x[n] are highly correlated when y[n] is shifted by -2, -1, 0 or 1 samples with respect to x[n].

The cross-correlation values in the second case are high when the lags are 4, 3, and 5, 2 ,1 and 0 indicating that y[n] and x[n] are highly correlated when y[n] is shifted by -4, -3, -2, -1, 0 or 1 samples with respect to x[n].

The cross-correlation values decrease as the lags increase and the sequences become less similar.

- Question 8:

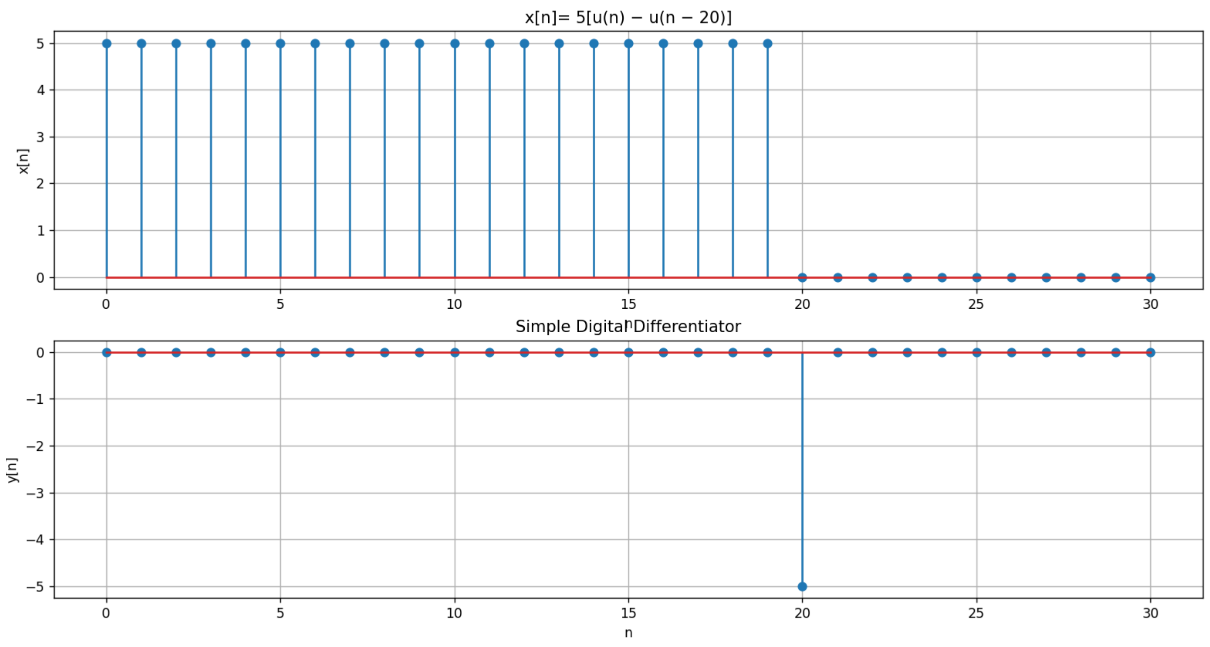
```
import numpy as np
from scipy.signal import lfilter
import matplotlib.pyplot as plt

a = [1, -1, 0.9]
b = [1]
n = np.arange(-20,121)

x = np.concatenate((np.zeros(20), [1], np.zeros(120)))
h = lfilter(b, a, x)

x2 = np.concatenate((np.zeros(20), np.ones(121)))
s = lfilter(b, a, x2)

plt.figure()
plt.subplot(2,1,1)
plt.stem(n, h)
plt.grid()
plt.title('impulse response')

plt.subplot(2,1,2)
plt.stem(n, s)
plt.grid()
plt.title('step response')
plt.show()
```



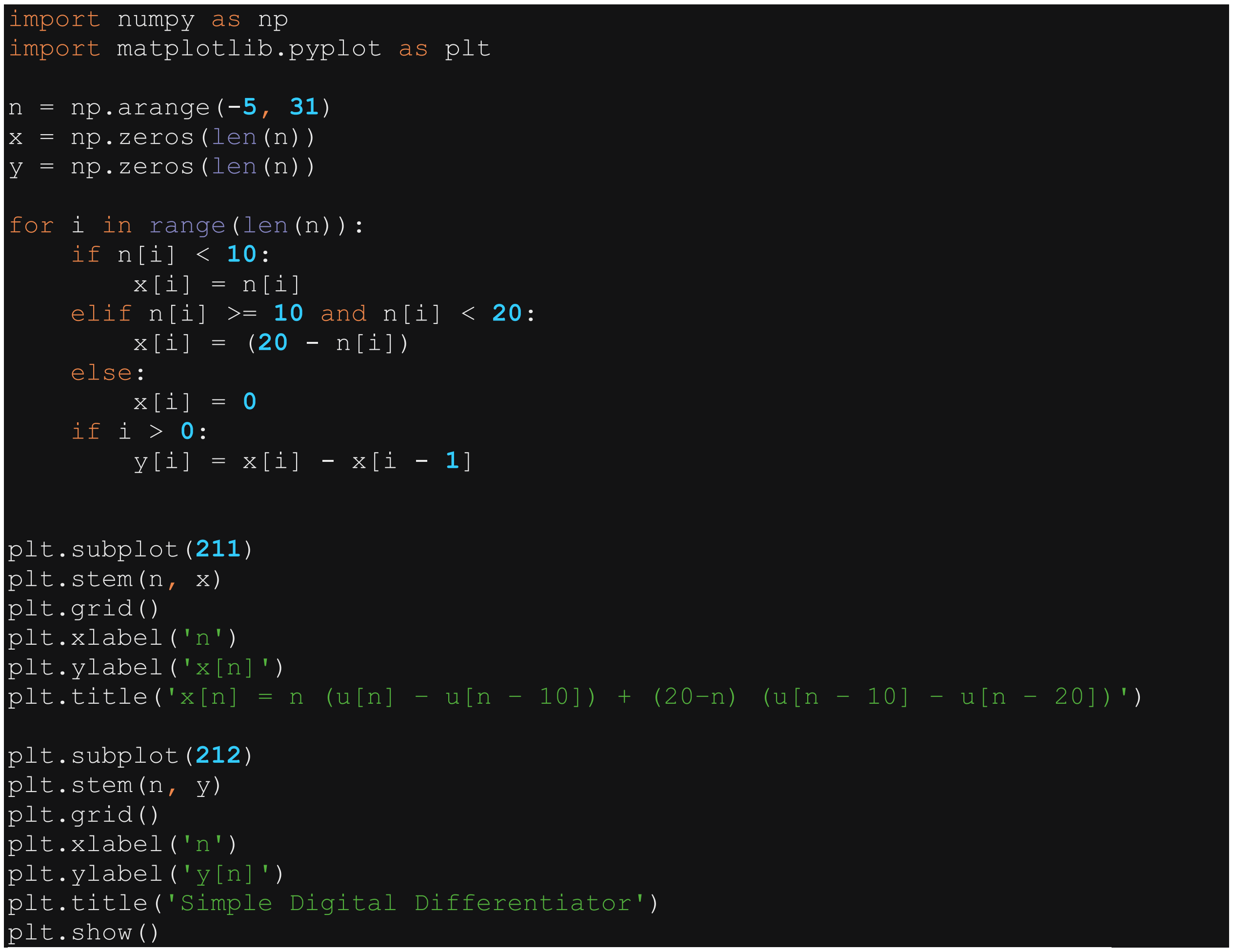
• Question 9:

A-

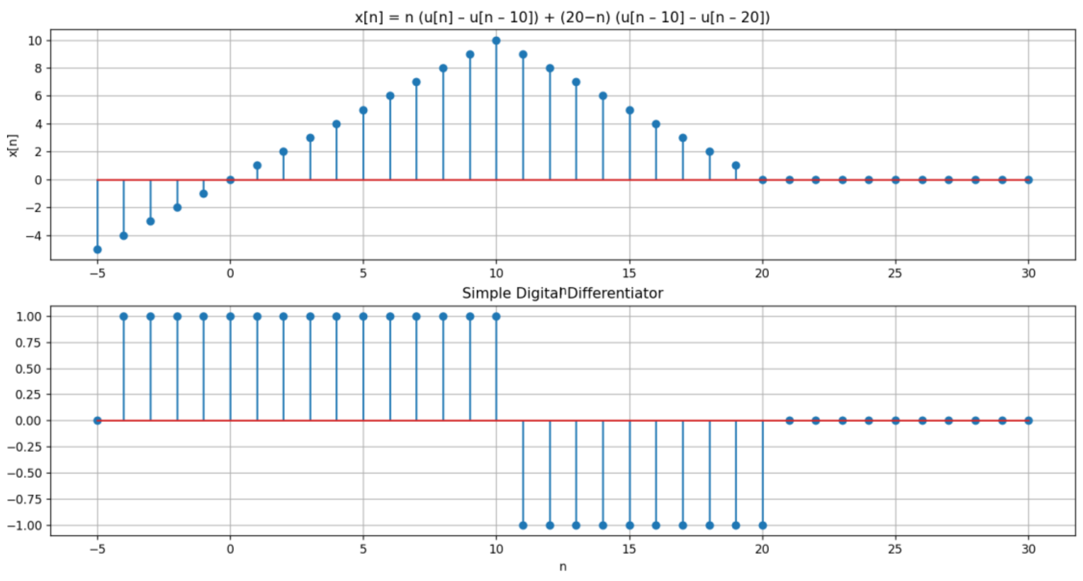
```
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(0, 31)
x = 5*(np.heaviside(n,1) - np.heaviside(n - 20,1))
y = np.zeros(len(x))

for i in range(1,len(x)):
    y[i] = x[i]-x[i-1]


plt.subplot(211)
plt.stem(n, x)
plt.grid()
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('x[n]= 5[u(n) − u(n − 20)]')

plt.subplot(212)
plt.stem(n, y)
plt.grid()
plt.xlabel('n')
plt.ylabel('y[n]')
plt.title('Simple Digital Differentiator')
plt.show()
```



B-

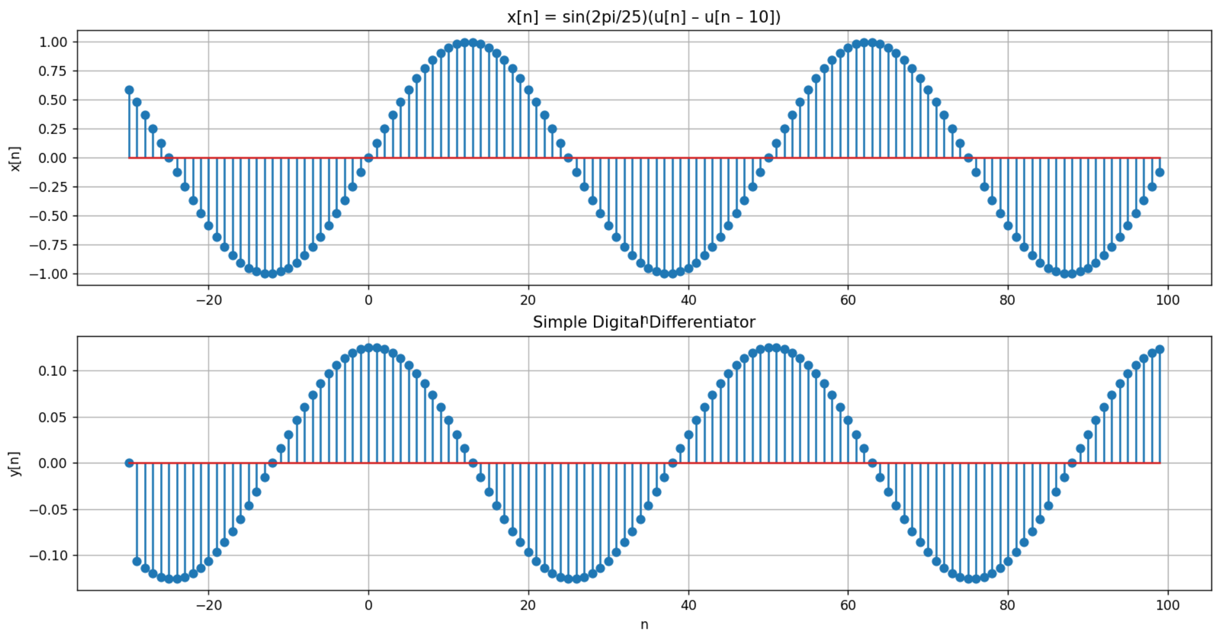


```python
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-5, 31)
x = np.zeros(len(n))
y = np.zeros(len(n))

for i in range(len(n)):
    if n[i] < 10:
        x[i] = n[i]
    elif n[i] >= 10 and n[i] < 20:
        x[i] = (20 - n[i])
    else:
        x[i] = 0
    if i > 0:
        y[i] = x[i] - x[i - 1]


plt.subplot(211)
plt.stem(n, x)
plt.grid()
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('x[n] = n (u[n] - u[n - 10]) + (20-n) (u[n - 10] - u[n - 20])')

plt.subplot(212)
plt.stem(n, y)
plt.grid()
plt.xlabel('n')
plt.ylabel('y[n]')
plt.title('Simple Digital Differentiator')
plt.show()
```



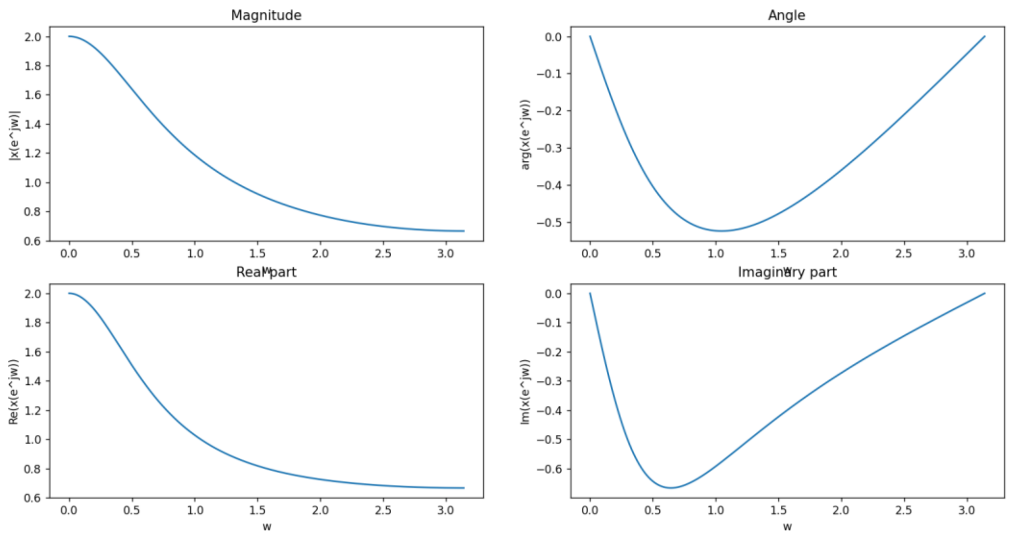
C-

```
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-30, 131)
x = np.zeros(len(n))
y = np.zeros(len(n))

for i in range(len(n)):
    if n[i] < 100:
        x[i] = np.sin((np.pi * n[i]) / 25)
    else:
        x[i] = 0
    if i > 0:
        y[i] = x[i] - x[i - 1]

plt.subplot(211)
plt.stem(n, x)
plt.grid()
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('x[n] = sin(2pi/25)(u[n] - u[n - 10])')

plt.subplot(212)
plt.stem(n, y)
plt.grid()
plt.xlabel('n')
plt.ylabel('y[n]')
plt.title('Simple Digital Differentiator')
plt.show()
```



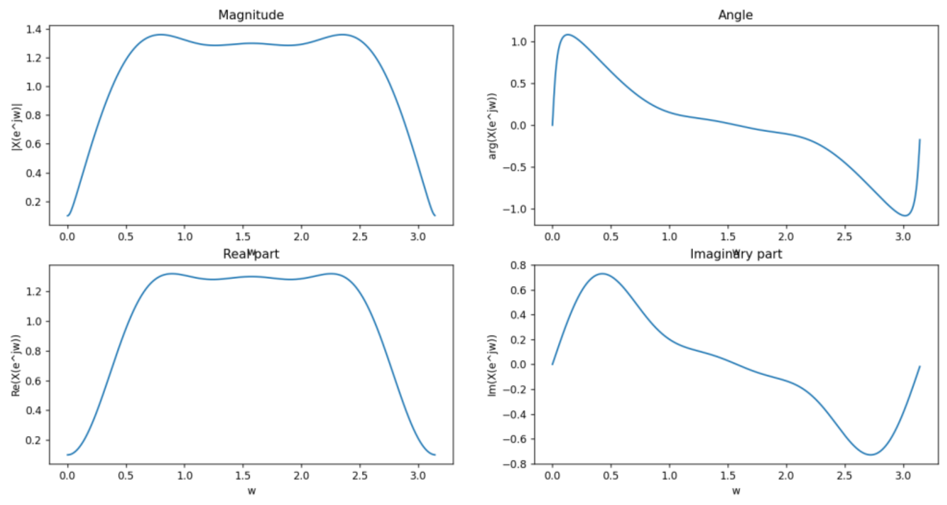
- Question 10:

```
import numpy as np
import matplotlib.pyplot as plt

w = np.linspace(0, np.pi, 501)
x = np.exp(1j*w)/(np.exp(1j*w) - 0.5)

plt.figure()
plt.subplot(2,2,1)
plt.plot(w, np.abs(x))
plt.xlabel('w')
plt.ylabel('|x(e^jw)|')
plt.title('Magnitude')

plt.subplot(2,2,2)
plt.plot(w, np.angle(x))
plt.xlabel('w')
plt.ylabel('arg(x(e^jw))')
plt.title('Angle')

plt.subplot(2,2,3)
plt.plot(w, np.real(x))
plt.xlabel('w')
plt.ylabel('Re(x(e^jw))')
plt.title('Real part')

plt.subplot(2,2,4)
plt.plot(w, np.imag(x))
plt.xlabel('w')
plt.ylabel('Im(x(e^jw))')
plt.title('Imaginary part')

plt.show()
```



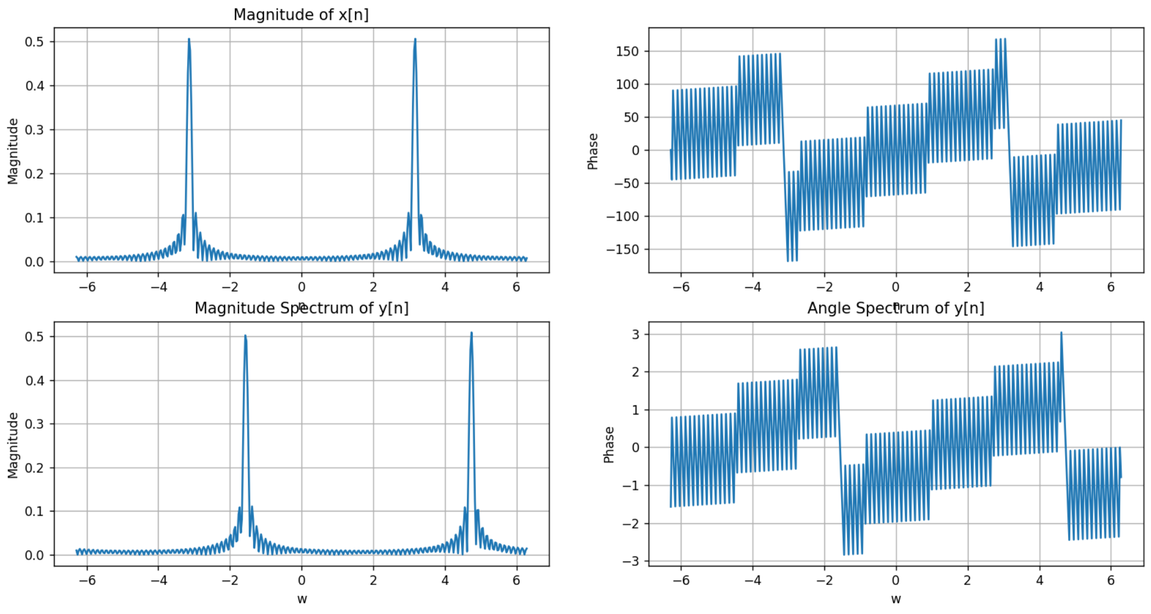
- Question 11:

```
import numpy as np
import matplotlib.pyplot as plt

x = [1, -0.5, -0.3, -0.1]

X = np.fft.fft(x, 501)

w = np.linspace(0, np.pi, 501)

plt.figure()
plt.subplot(2,2,1)
plt.plot(w, np.abs(X))
plt.xlabel('w')
plt.ylabel('|X(e^jw)|')
plt.title('Magnitude')

plt.subplot(2,2,2)
plt.plot(w, np.angle(X))
plt.xlabel('w')
plt.ylabel('arg(X(e^jw))')
plt.title('Angle')

plt.subplot(2,2,3)
plt.plot(w, np.real(X))
plt.xlabel('w')
plt.ylabel('Re(X(e^jw))')
plt.title('Real part')

plt.subplot(2,2,4)
plt.plot(w, np.imag(X))
plt.xlabel('w')
plt.ylabel('Im(X(e^jw))')
plt.title('Imaginary part')

plt.show()
```



- Question 12:

```
import numpy as np
import matplotlib.pyplot as plt

L = 100
w = np.linspace(-2*np.pi, 2*np.pi, 401)

n = np.arange(0, L+1)
x = np.cos(np.pi*n/2)

X = (1/L)*np.fft.fft(x, len(w))

y = np.exp(1j*np.pi*n/4)*x

Y = (1/L)*np.fft.fft(y, len(w))

plt.subplot(2,2,1)
plt.plot(w, np.abs(X))
plt.grid()
plt.xlabel('n')
plt.ylabel('Magnitude')
plt.title('Magnitude of x[n]')

plt.subplot(2,2,2)
plt.plot(w, (180/np.pi)*np.angle(X))
plt.grid()
plt.xlabel('n')
plt.ylabel('Phase')

plt.subplot(2,2,3)
plt.plot(w, np.abs(Y))
plt.xlabel('w')
plt.ylabel('Magnitude')
plt.title('Magnitude Spectrum of y[n]')

plt.subplot(2,2,4)
plt.plot(w, np.angle(Y))
plt.xlabel('w')
plt.ylabel(Phase')
plt.title('Angle Spectrum of y[n]')

plt.show()
```



Multiplying a signal with a complex exponential signal with a certain frequency will shift the phase of the signal by the same amount as the frequency of the complex exponential signal. As can be noticed the plot above the two plots in the top row are similar to the one in the bottom row but they are shifted by pi/4.