
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Digital Signal Processing

ENCS4310

Assignment (1)

---

Prepared By: Student Name: Ibrahim Nobani

Student ID: 1190278

Instructor: Dr. Qadri Mayyala

Section: 2

Date: 12/1/2022

---

Q1)

Q1. Generate and plot each of the following sequences over the indicated interval.

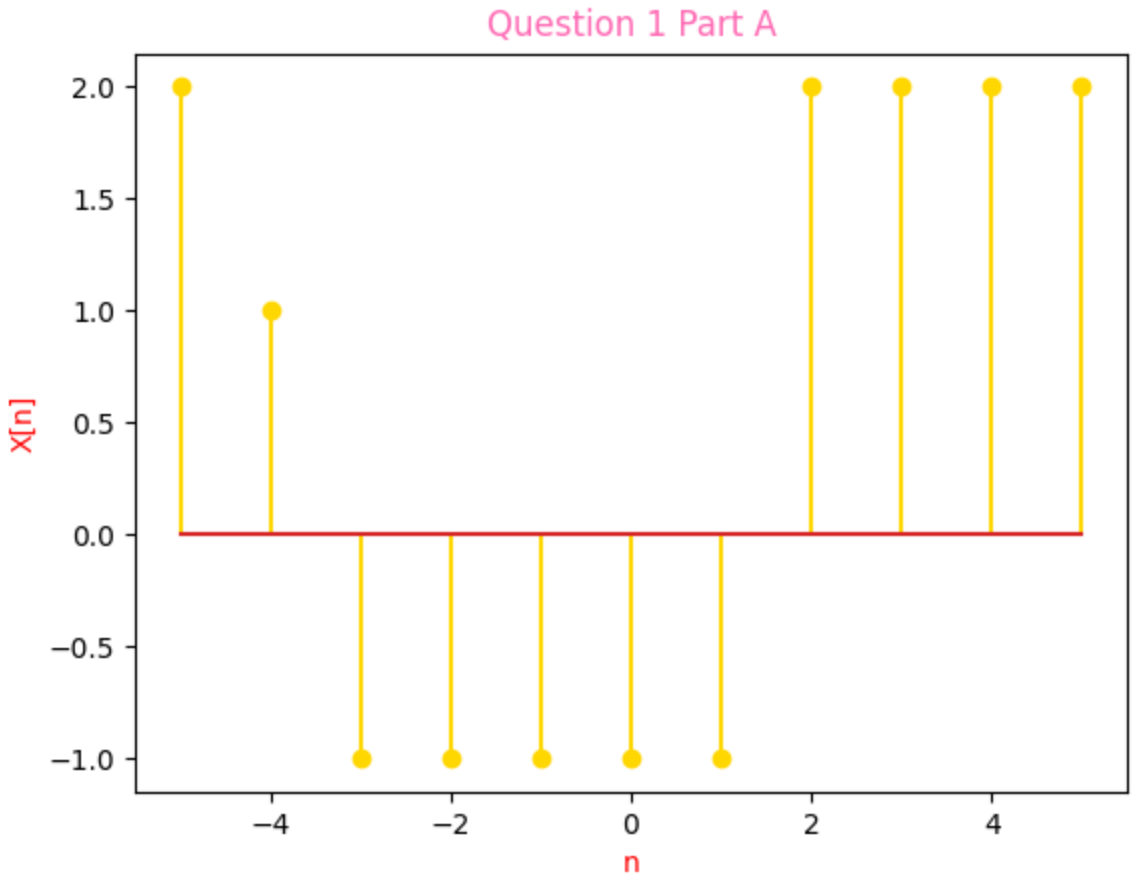
A)

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

Code:

```
import numpy as np
import matplotlib.pyplot as plt
n = np.linspace(-5, 5, 11) #Identify Interval
u = np.heaviside(n,1) #The unit sample sequence
x = 2*np.roll(u,2) - np.roll(u,-4) # This does the sequence shift
plt.stem(n, x,'gold')
plt.title("Question 1 Part A",color = 'hotpink')
plt.xlabel('n',color = 'red')
plt.ylabel('X[n]',color = 'red')
plt.show()
```

Output:

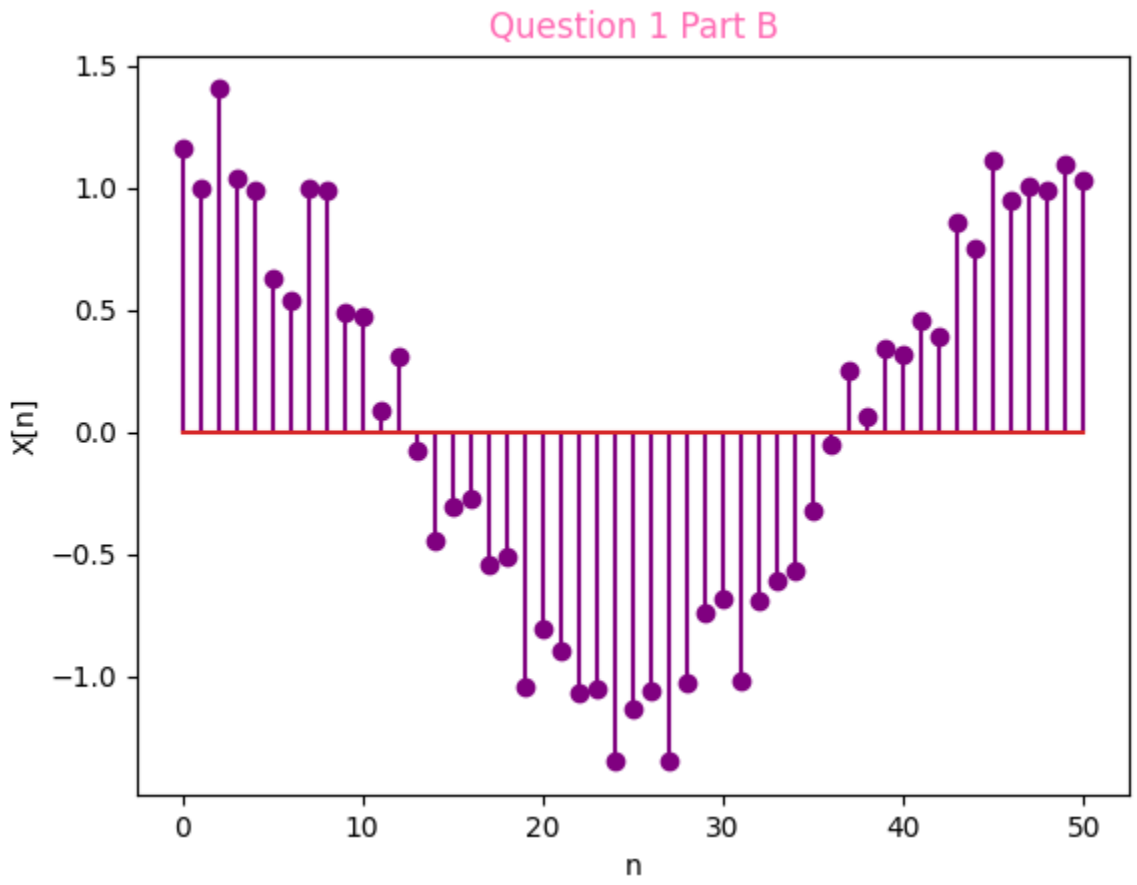


B)

$y[n] = cos(0.04\pi n) + 0.2w(n)$, $0 \leq n \leq 50$. where w(n) is a Gaussian random sequence with zero mean and unit variance

Code:

```
import numpy as np
import matplotlib.pyplot as plt
n2 = np.linspace(0, 50, 51)
Xn = np.cos(0.04*np.pi*n2)
Ws = np.random.normal(0, 1, 51) #eng.randn(1, 51)
Yn = Xn + 0.2*Ws
plt.stem(n2, Yn,'purple')
plt.title("Question 1 Part B",color = 'hotpink')
plt.xlabel('n')
plt.ylabel('X[n]')
plt.show()
```

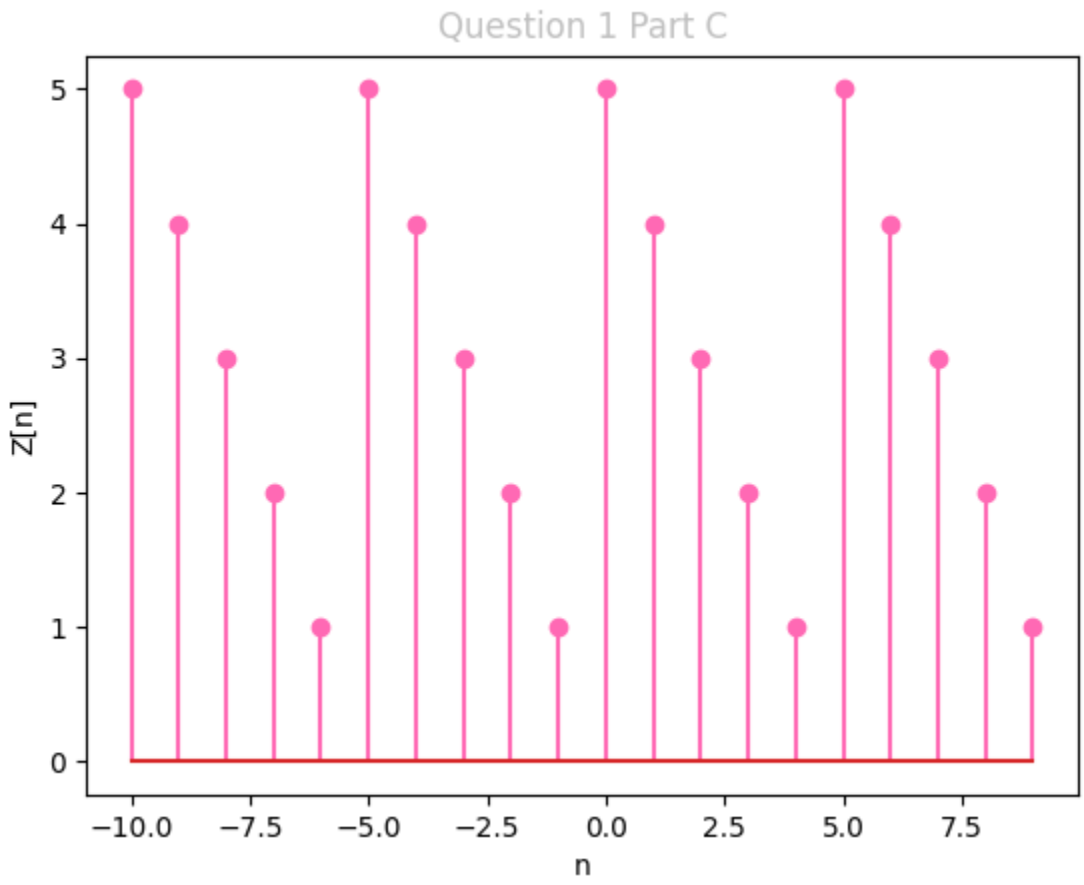


C)

C) $z[n] = \{..., 5,4,3,2,1,\underline{5},4,3,2,1,5,4,3,2,1, ...\}; \ -10 \leq n \leq 9,$

```
import numpy as np
import matplotlib.pyplot as plt
n = np.linspace(-10, 9, 20)
a = np.array([5,4,3,2,1])
z = np.tile(a,4)
plt.stem(n, z,'hotpink')
plt.title("Question 1 Part C",color = 'silver')
plt.xlabel('n')
plt.ylabel('Z[n]')
plt.show()
```

Output:



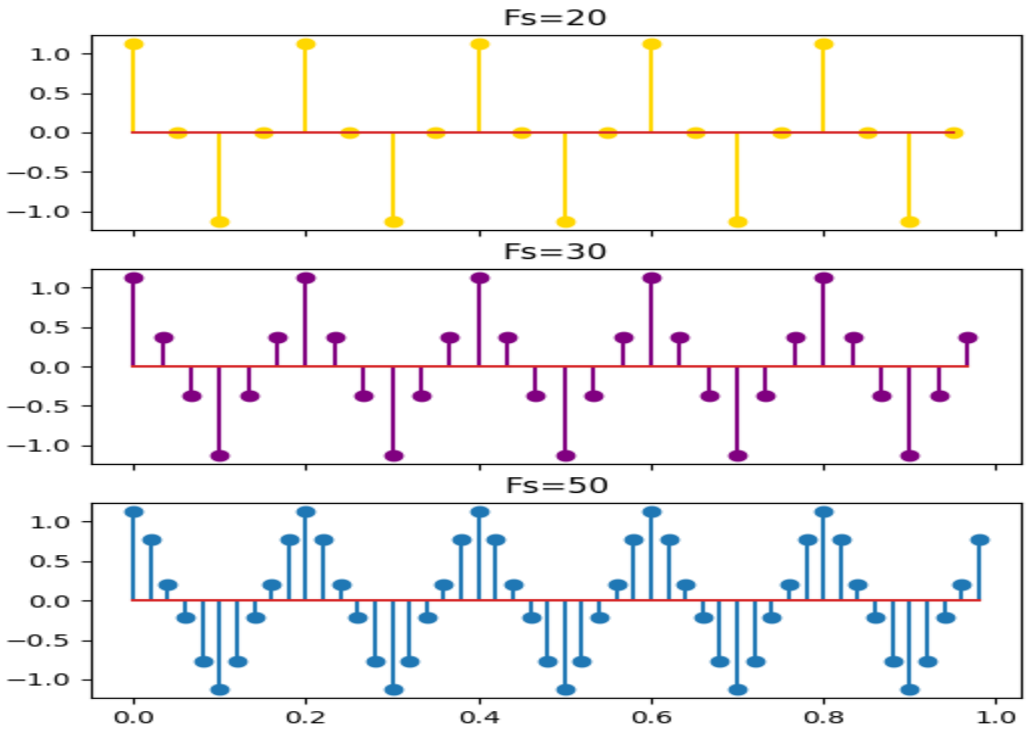
Q2. Generate and plot each of the following sequences over the indicated interval.

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

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

Code:

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

# The time intervals.
tn1 = np.arange(0, 1, 1/20)
tn2 = np.arange(0, 1, 1/30)
tn3 = np.arange(0, 1, 1/50)
f1,f2=5,15
Gn1 = np.cos(2*np.pi*f1*tn1) + 0.125*np.cos(2*np.pi*f2*tn1)
Gn2 = np.cos(2*np.pi*f1*tn2) + 0.125*np.cos(2*np.pi*f2*tn2)
Gn3 = np.cos(2*np.pi*f1*tn3) + 0.125*np.cos(2*np.pi*f2*tn3)
fig, axis = plt.subplots(3, 1, figsize=(6, 8), sharex=True, sharey=True)
axis[0].stem(tn1, Gn1,'gold')
axis[0].set_title('Fs=20')
axis[1].stem(tn2, Gn2,'purple')
axis[1].set_title('Fs=30')
axis[2].stem(tn3, Gn3)
axis[2].set_title('Fs=50')
plt.show()
```



The sampling frequency Fs and the signal frequency Fm have a relation that states that

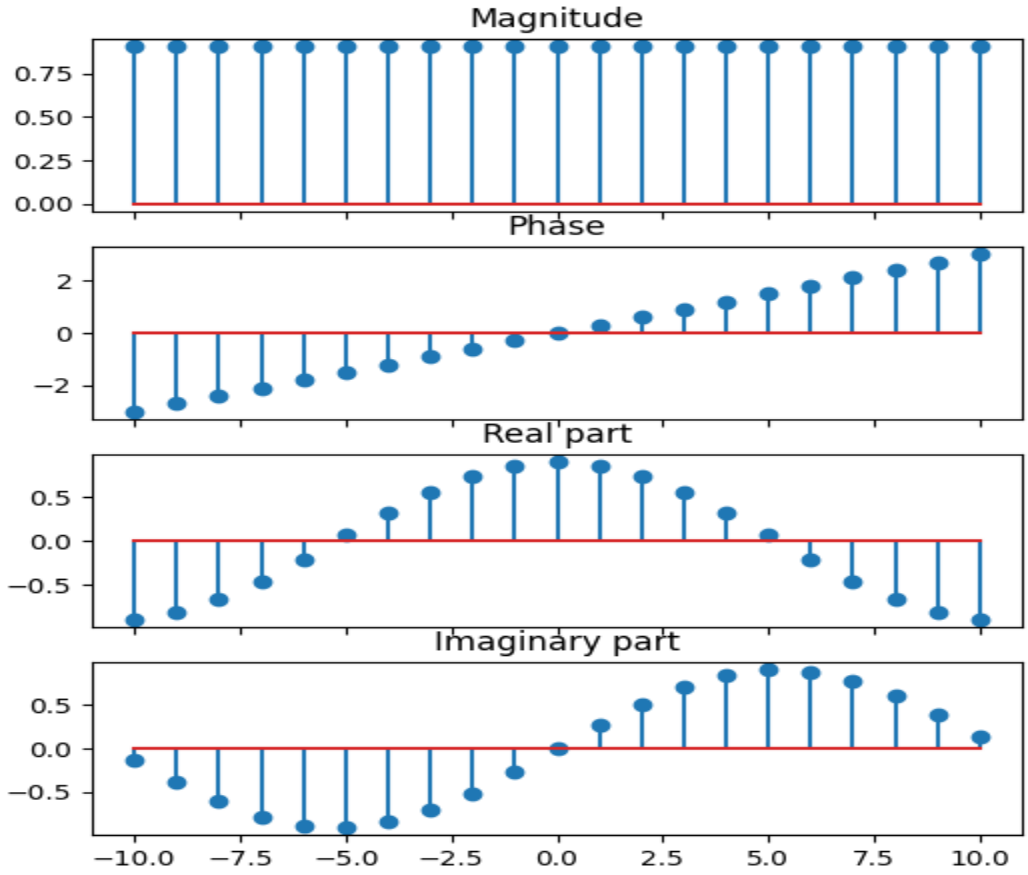
Fs >= Fm The discrete samples that are collected from the signal are impacted by the varied sampling rates. When Fs = 50Hz which is much bigger than 2Fm the signal seems more precise and smooth when the sampling rate is higher because more samples are taken per second. The second when Fs=30Hz which is 2Fm the system respond is not that good, while when Fs=20Hz which is lower than 2Fm this makes it seem more discrete and less precise when the sampling rate is decreased because fewer samples are taken each second.

Q3. Generate the complex-valued signal

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

and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots.

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

n = np.arange(-10, 11)
X = np.exp(-0.1 + 1j*0.3*n)
fig, axs = plt.subplots(4, 1, figsize=(6, 8), sharex=True)
# Mangitude using abs in np
axs[0].stem(n, np.abs(X))
axs[0].set_title('Magnitude')
# Phase using angle in np
axs[1].stem(n, np.angle(X))
axs[1].set_title('Phase')
# Real using real in np
axs[2].stem(n, np.real(X))
axs[2].set_title('Real part')
# Imaginary using imag in numpy
axs[3].stem(n, np.imag(X))
axs[3].set_title('Imaginary part')
plt.show()
```



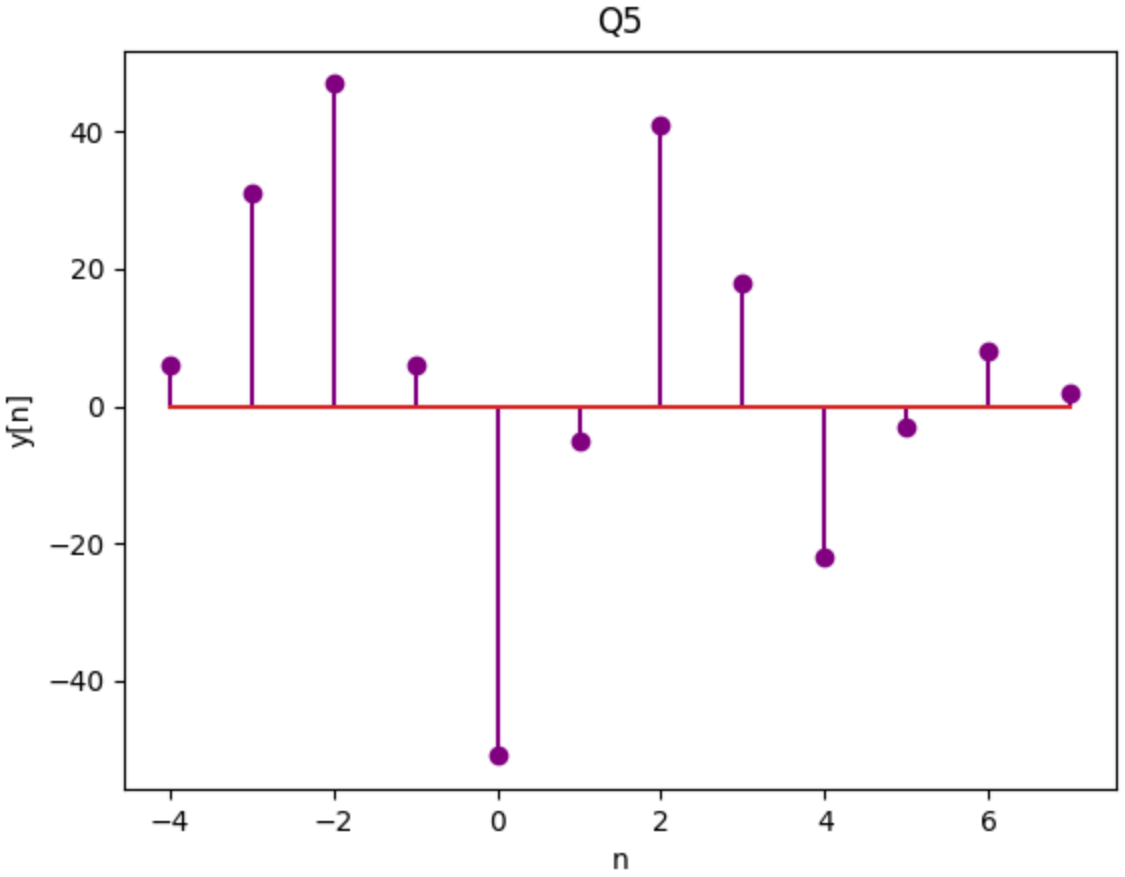

Q5. For

$$x[n] = [3, 11, 7, 0, -1, 4, 2], \quad -3 \le n \le 3;$$

$$h[n] = [2, 3, 0, -5, 2, 1], \quad -1 \le n \le 4$$

Find and plot y[n].

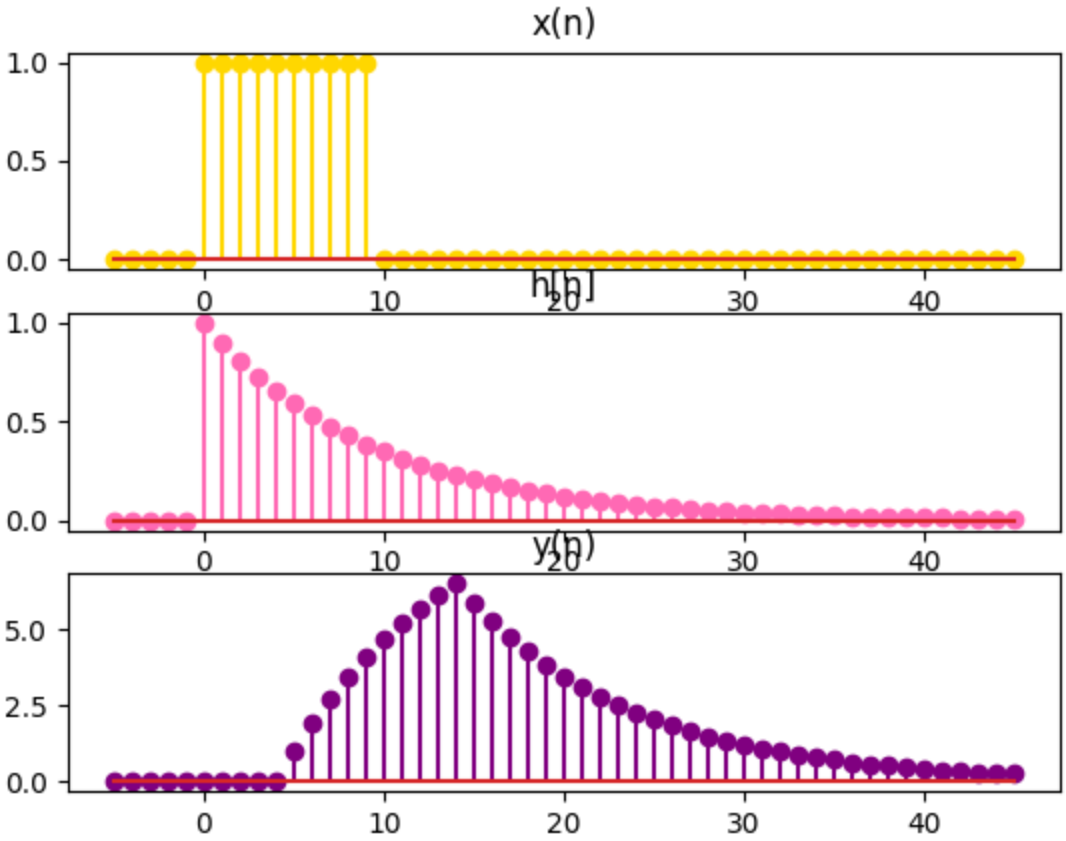
```
import matplotlib.pyplot as plt
import numpy as np

x = [3, 11, 7, 0, -1, 4, 2]
h = [2, 3, 0, -5, 2, 1]
y = np.convolve(x, h)
n1 = np.linspace(-3, 3, len(x))
n2 = np.linspace(-1, 4, len(h))
n3 = np.linspace(n1[0] + n2[0], max(n1) + max(n2), len(y))

plt.stem(n3, y, "purple")
plt.xlabel("n")
plt.ylabel("y[n]")
plt.title("Q5")
plt.show()
```




Q6. Let the rectangular pulse x(n) = u(n) − u(n − 10) be an input to an LTI system with impulse response $h[n] = (0.9)^n u(n)$

Plot x{n], h[n], Find and plot the output y(n). Consider the interval [-5, 45].

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

n = np.arange(-5, 46) #Interval.
Xn = (np.heaviside(n,1) - np.heaviside(n - 10,1))
hn = (0.9) ** n * np.heaviside(n,1)
Yn = np.convolve(Xn, hn)
fig, axis = plt.subplots(3, 1, figsize=(10,5))
axis[0].stem(n, Xn,"gold")
axis[0].set_title('x(n)')
axis[1].stem(n, hn,"hotpink")
axis[1].set_title('h[n]')
axis[2].stem(n, Yn[:51],"purple")
axis[2].set_title('y(n)')
axis[2].set_xlabel('n')
plt.show()
```




Q7. To demonstrate one application of the crosscorrelation sequence.

Let $x[n] = [3,11,7,\underline{0},-1,4,2]$ be a prototype sequence,

A) let y(n) be its noise-corrupted-and-shifted version

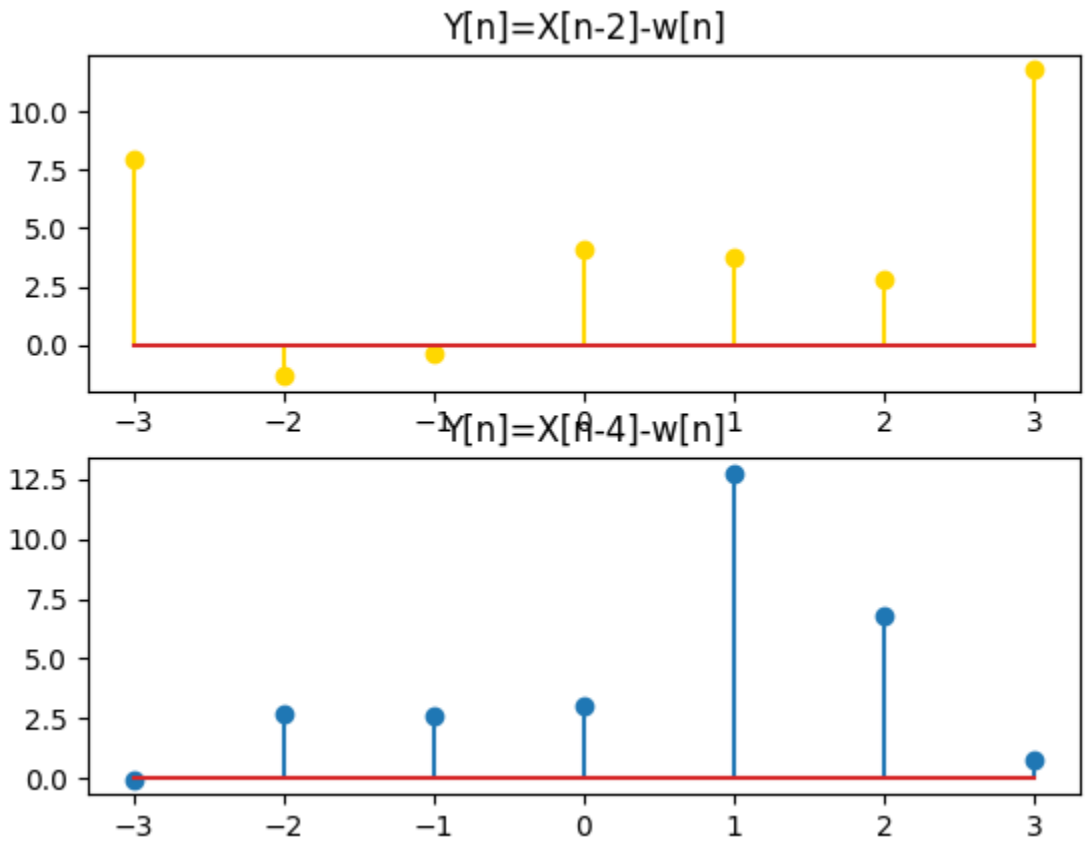
$$y[n]=x[n-2]+w[n]$$

where w[n] is Gaussian sequence with mean 0 and variance 1. Compute the crosscorrelation between y[n] and x[n] and comment on the results.

B) Repeat part (a) for $y[n]=x[n-4]+w[n]$

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

n = np.arange(-3, 4)
Xn = np.array([3, 11, 7, 0, -1, 4, 2])

n2 = np.random.normal(0, 1, size=len(n))
Yn1 = np.roll(Xn, -2) + n2
Yn2 = np.roll(Xn, -4) + n2
fig,axis = plt.subplots(2,1)
axis[0].stem(n, Yn1,'gold')
axis[0].set_title("Y[n]=X[n-2]-w[n]")
axis[1].stem(n, Yn2)
axis[1].set_title("Y[n]=X[n-4]-w[n]")
```

```
plt.show()

corr = np.corrcoef(Xn, Yn1)[0, 1]
corr2 = np.corrcoef(Xn, Yn2)[0, 1]
print(corr)
print(corr2)
```




Q8. Given the following difference equation

$$y[n]-y[n-1]+0.9y[n-2] = x[n]$$

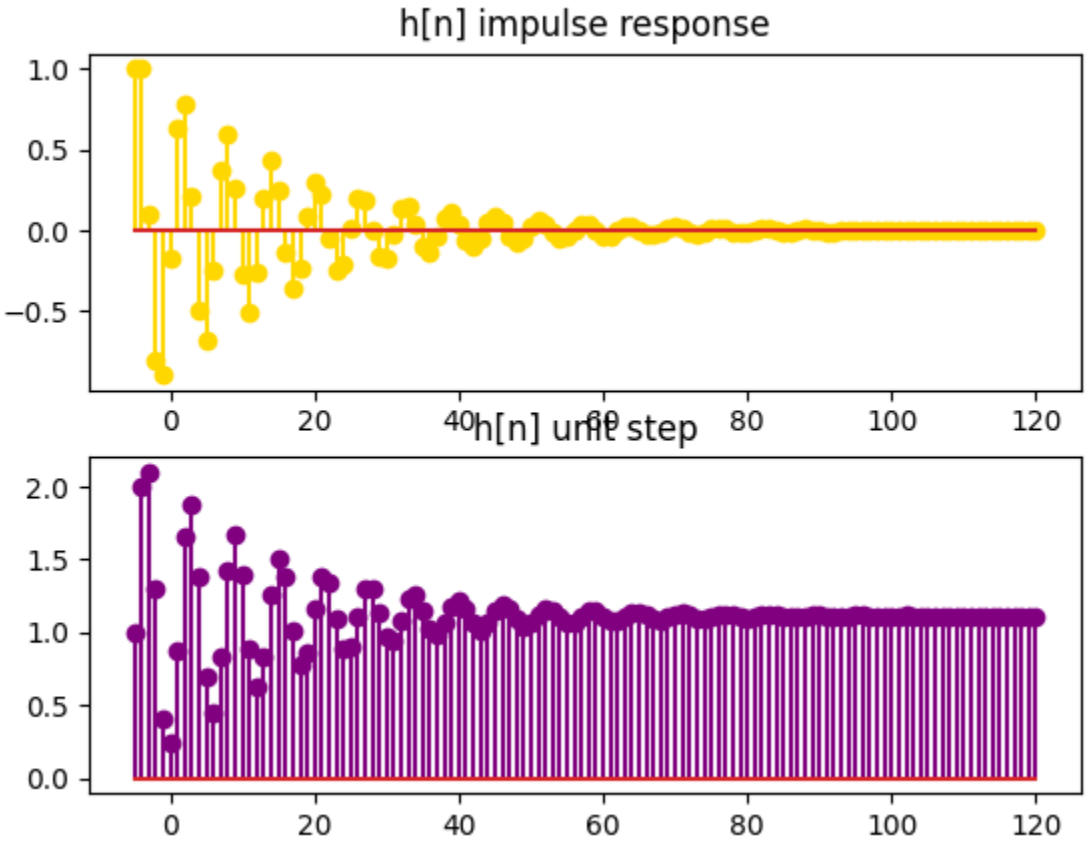
A) Calculate and plot the impulse response h(n) at n = −5,..., 120.

B) Calculate and plot the unit step response s(n) at n = −5,..., 120.

C) Is the system specified by h(n) stable?

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

Ycoef = [1, -1, 0.9]
Xcoef = [1]
n = np.linspace(-5, 120, 126)
#Impulse
```

```
Xn = np.zeros(len(n))
Xn[0] = 1
hn = lfilter(Xcoef, Ycoef, Xn)
# Unit step
Xb = np.ones(len(n))
fig, axis = plt.subplots(2, 1)
hnb = lfilter(Xcoef, Ycoef, Xb)
print(sum(hn))
axis[0].stem(n, hn,'gold')
axis[0].set_title('h[n] impulse response')
axis[1].stem(n, hnb,'purple')
axis[1].set_title('h[n] unit step')
plt.show()
```



Q9. A "simple" digital differentiator is given by

$$y[n]=x[n]-x[n-1]$$

which computes a backward first-order difference of the input sequence. Implement this differentiator on the following sequences, and plot the results. Comment on the appropriateness of this simple differentiator.

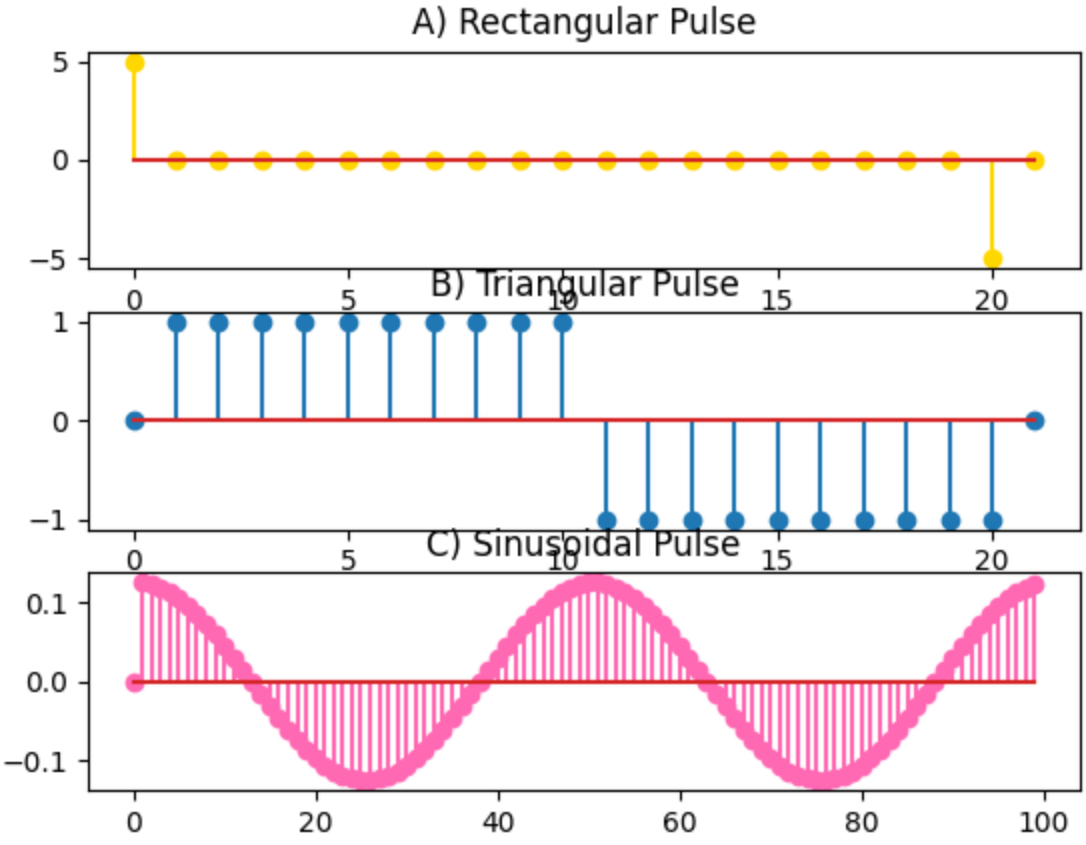
A) Rectangular pulse: **x[n]= 5[u(n) − u(n − 20)]**

B) Triangular pulse: **x[n] = n (u[n] − u[n − 10]) + (20−n) (u[n − 10] − u[n − 20])**

C) Sinusoidal pulse: $x[n] = sin\left(\frac{\pi n}{25}\right)(u[n]-u[n-100])$

---

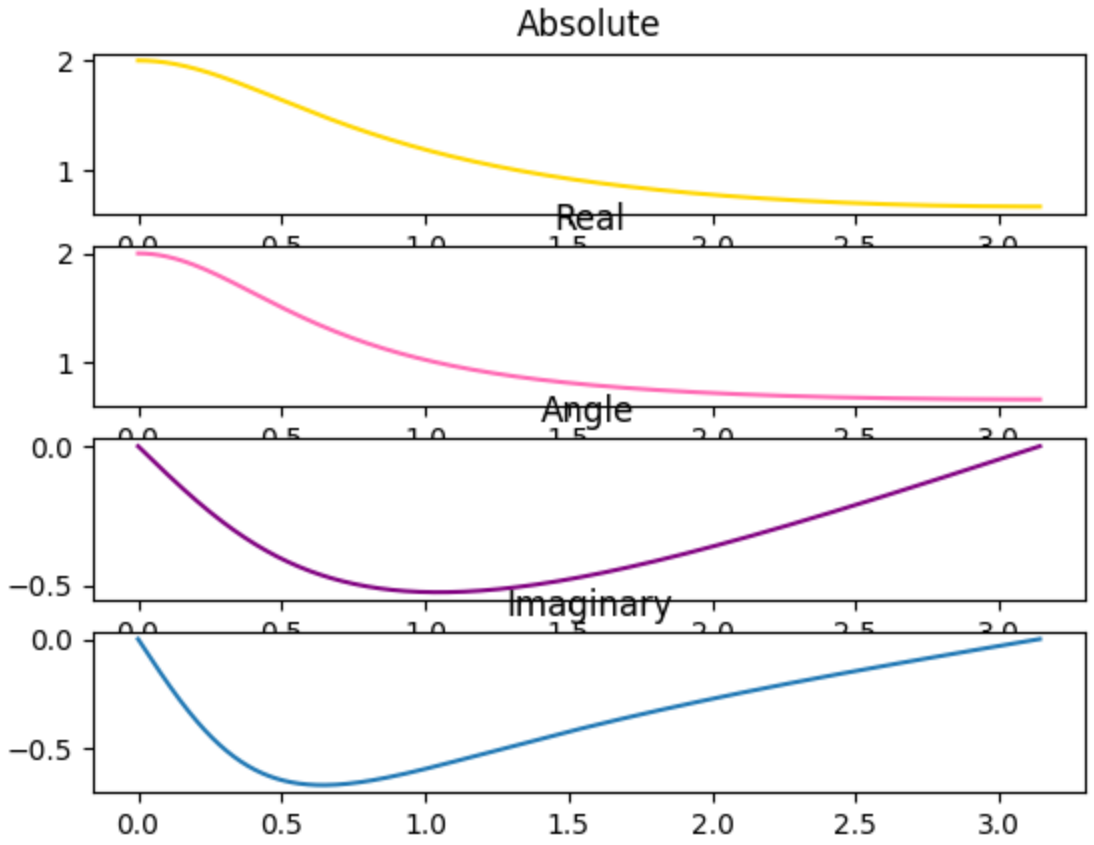
```
import numpy as np
from scipy.signal import lfilter
import matplotlib.pyplot as plt

a = [1]
b = [1, -1]
n1 = np.linspace(0, 21, 22)
###### Part A
Xn_A = 5 * (np.heaviside(n1, 1) - np.heaviside(n1 - 20, 1))
Yn_A = lfilter(b, a, Xn_A)
###### Part B
Xn_B= n1*(np.heaviside(n1, 1)-np.heaviside(n1 - 10, 1))+(20 -
n1)*(np.heaviside(n1 - 10, 1)- np.heaviside(n1 - 20, 1))
Yn_B = lfilter(b, a, Xn_B)
###### Part C
n2 = np.arange(0, 100)
Xn_C = np.sin((np.pi * n2) / 25)*(np.heaviside(n2, 1)-np.heaviside(n2 - 100,
1))
Yn_C = lfilter(b, a, Xn_C)

fig, axis = plt.subplots(3, 1)
axis[0].stem(n1, Yn_A,'gold')
axis[0].set_title("A) Rectangular Pulse")
axis[1].stem(n1, Yn_B)
axis[1].set_title("B) Triangular Pulse")
axis[2].stem(n2, Yn_C,'hotpink')
axis[2].set_title("C) Sinusoidal Pulse")
plt.show()
```




Q10. For $\mathbf{x[n] = (0.5)^n\ u[n]}$. The corresponding DTFT is $X(e^{j\omega}) = \dfrac{e^{j\omega}}{e^{j\omega} - 0.5}$.

Evaluate $X(e^{j\omega})$ at 501 equispaced points between $[0,\pi]$ and plot its magnitude, angle, real, and imaginary parts.

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

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

fig, axis = plt.subplots(4, 1)
axis[0].plot(n, np.abs(Xn), 'gold')
axis[0].set_title('Absolute')
axis[1].plot(n, np.real(Xn), 'hotpink')
axis[1].set_title('Real')
axis[2].plot(n, np.angle(Xn), 'purple')
axis[2].set_title('Angle')
axis[3].plot(n, np.imag(Xn))
axis[3].set_title('Imaginary')
plt.show()
```



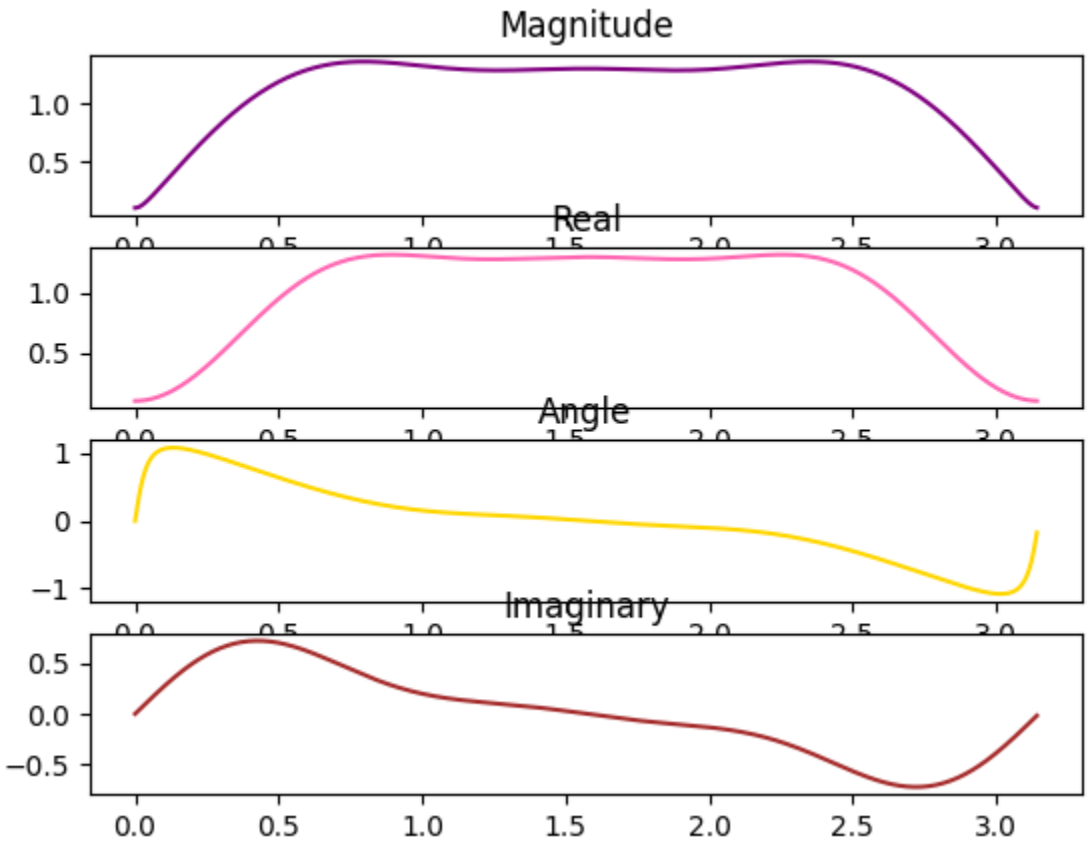

Q.11 Consider the sequence $x[n]=\{1,-0.5,-0.3,-0.1\}$

A) Numerically compute the discrete-time Fourier transform of at 501 equispaced frequencies between $[0,\pi]$.

B) plot its magnitude, angle, real, and imaginary parts.

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

Xn = [1, -0.5, -0.3, -0.1]
Xf = np.fft.fft(Xn, 501)
n = np.linspace(0, np.pi, 501)

fig, axis = plt.subplots(4, 1)
axis[0].plot(n, np.abs(Xf), 'purple')
axis[0].set_title('Magnitude')

axis[1].plot(n, np.real(Xf), 'hotpink')
axis[1].set_title('Real')

axis[2].plot(n, np.angle(Xf), 'gold')
axis[2].set_title('Angle')
```

```
axis[3].plot(n, np.imag(Xf), 'brown')
axis[3].set_title('Imaginary')
plt.show()
```




Q.12 Let $x[n]=\cos(\frac{\pi n}{2})$, $0 \le n \le 100$ and $y[n]=e^{j\pi n/4}x[n]$

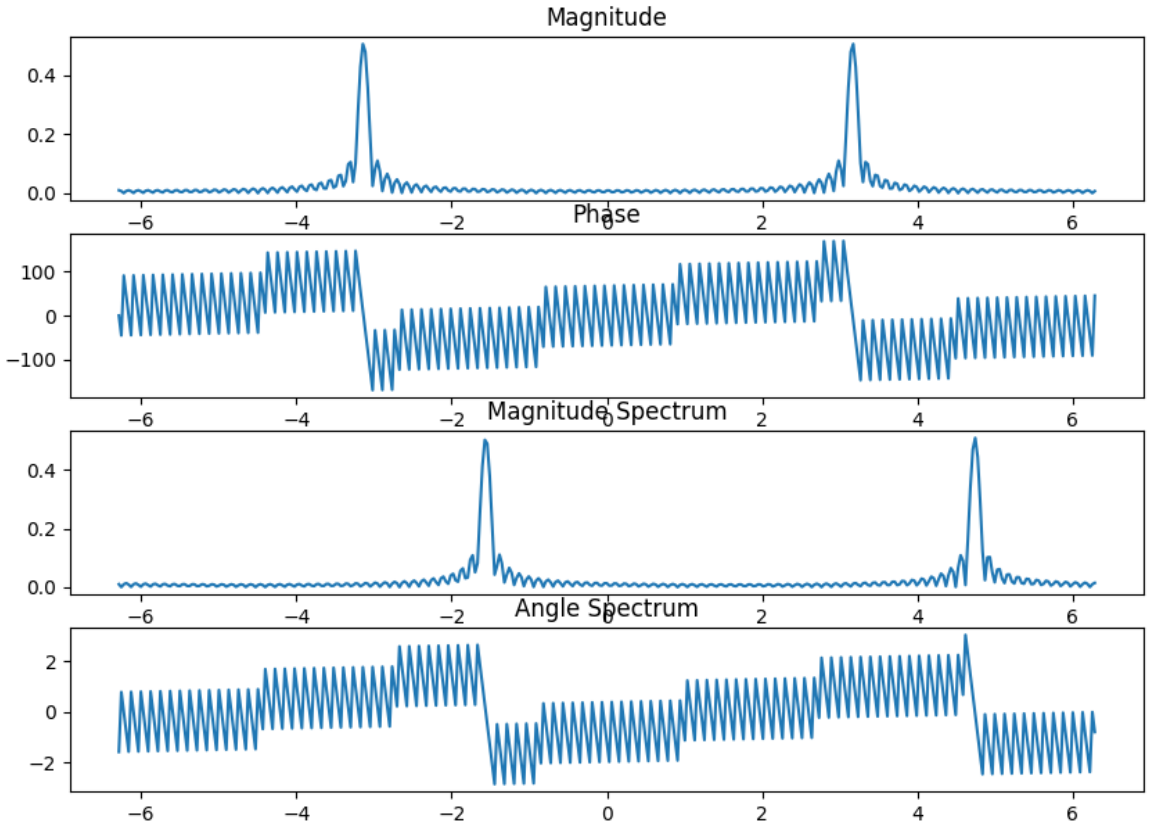
A) Numerically compute the discrete-time Fourier transform of at 401 equispaced frequencies between $[-2\pi,2\pi]$.

B) plot its magnitude, angle spectrum.

C) Comment on the relation between x[n] and y[n].

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

n1 = np.linspace(-2*np.pi, 2*np.pi, 401)
n2 = np.arange(0, 101)
Xn = np.cos(np.pi*n2/2)
Xf = (1/100)*np.fft.fft(Xn, len(n1))
yn = np.exp(1j*np.pi*n2/4)*Xn
Yf = (1/100)*np.fft.fft(yn, len(n1))
```

```
fig, axis = plt.subplots(4, 1,figsize=(10, 10))
axis[0].plot(n1, np.abs(Xf))
axis[0].set_title('Magnitude')

axis[1].plot(n1, (180/np.pi)*np.angle(Xf))
axis[1].set_title('Phase')

axis[2].plot(n1, np.abs(Yf))
axis[2].set_title('Magnitude Spectrum')

axis[3].plot(n1, np.angle(Yf))
axis[3].set_title('Angle Spectrum')
plt.show()
```




When a signal is multiplied by a complex exponential signal at a particular frequency, the phase of the original signal is shifted by the same quantity. The two plots in the top row of the figure above are similar to the one in the bottom row, except they are displaced by pi/4, as can be seen.