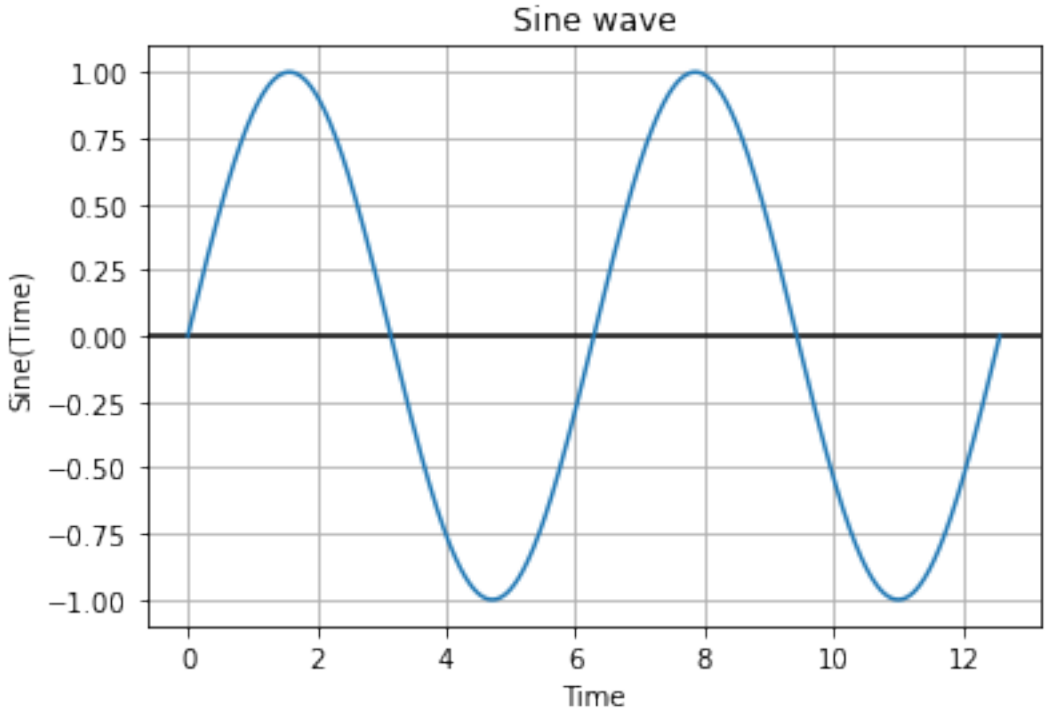
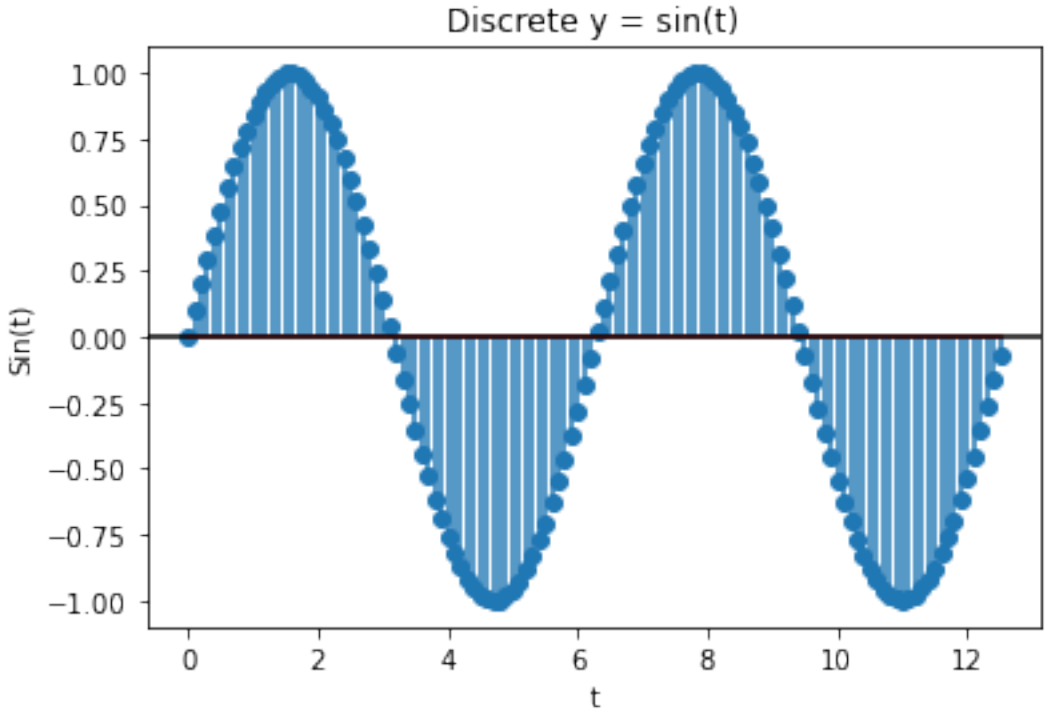
```
#Q1 (a)    Odd Signal - Sine Wave

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x = np.arange(0,4*np.pi,0.001)    #this line makes an array from 0 to
4pi, with a step of 0.001
y = np.sin(x)                     #this calculates sin of all values
in our array

plt.title('Sine wave')            #this line gives the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Sine(Time)')          #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph

plt.show()                        #this line shows the graph and
successfully terminates the program
```



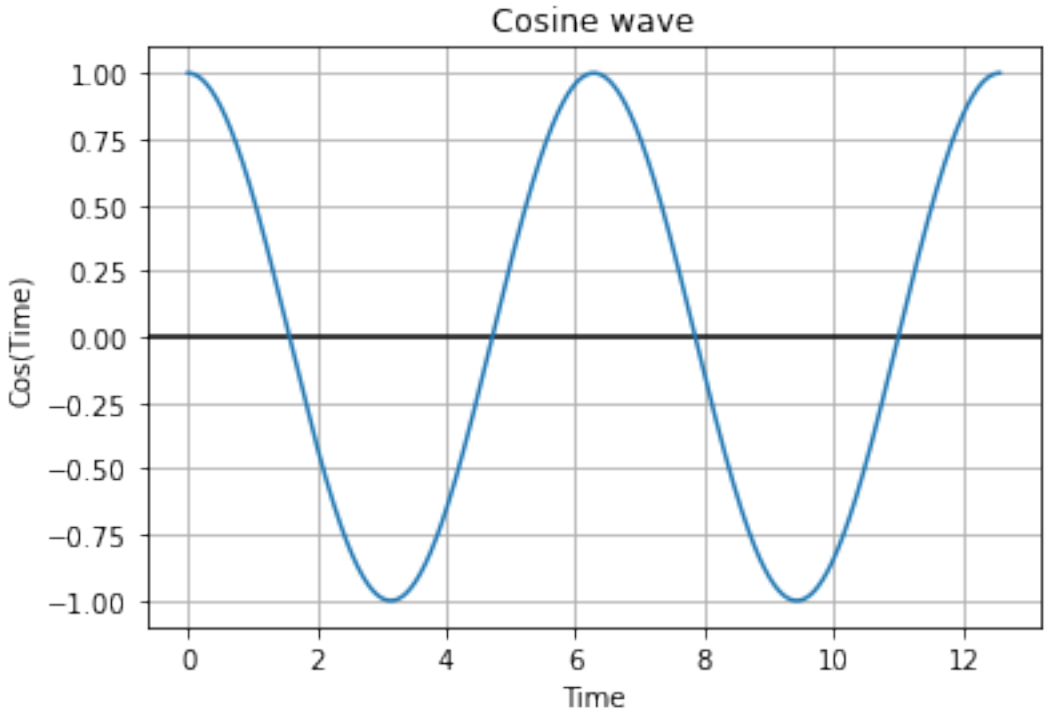

```
#Q1 (a) Odd Signal - DISCRETE
```

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

x = np.arange(0,4*np.pi,0.1)
y = np.sin(x)
plt.stem(x,y,use_line_collection='true' )
plt.title('Discrete y = sin(t)')
plt.xlabel('t')
plt.ylabel('Sin(t)')
plt.axhline(y=0,color='k')
plt.show()
```

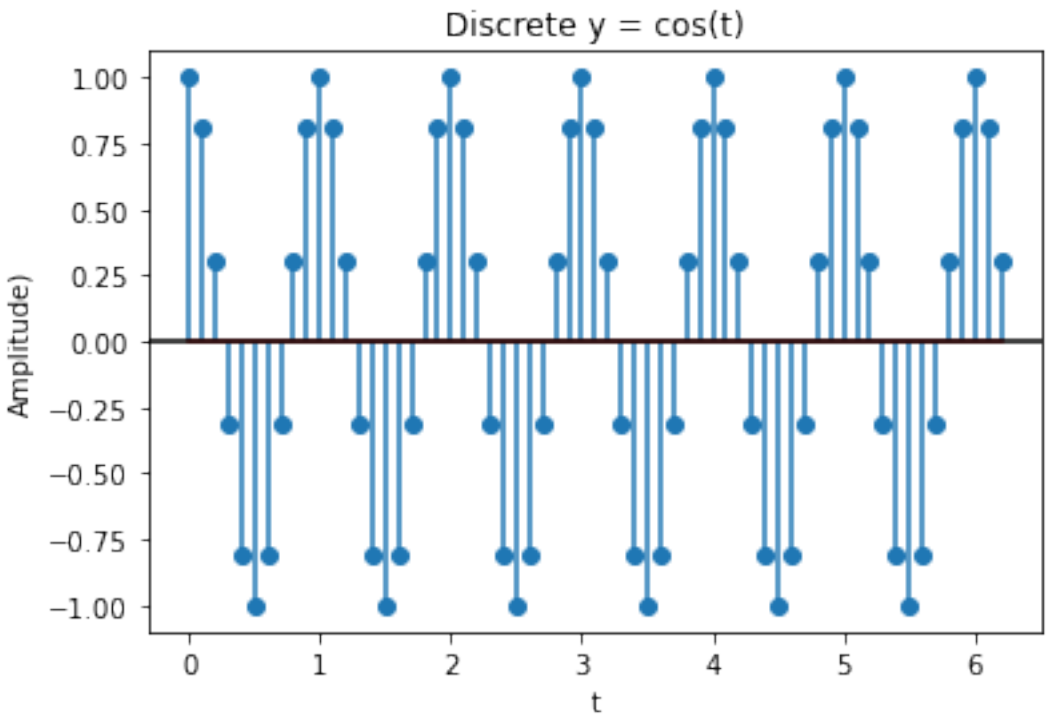



```
#Q1 (b)    Even Signal - Cosine Wave

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x = np.arange(0,4*np.pi,0.001)    #this line makes an array from 0 to
4pi, with a step of 0.001
y = np.cos(x)                     #this calculates cos of all values
in our array

plt.title('Cosine wave')          #this line gives the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Cos(Time)')           #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
```

```
line at y=0, with color code as black

plt.grid()                          #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                       #this line plots the graph

plt.show()                          #this line shows the graph and
successfully terminates the program
```

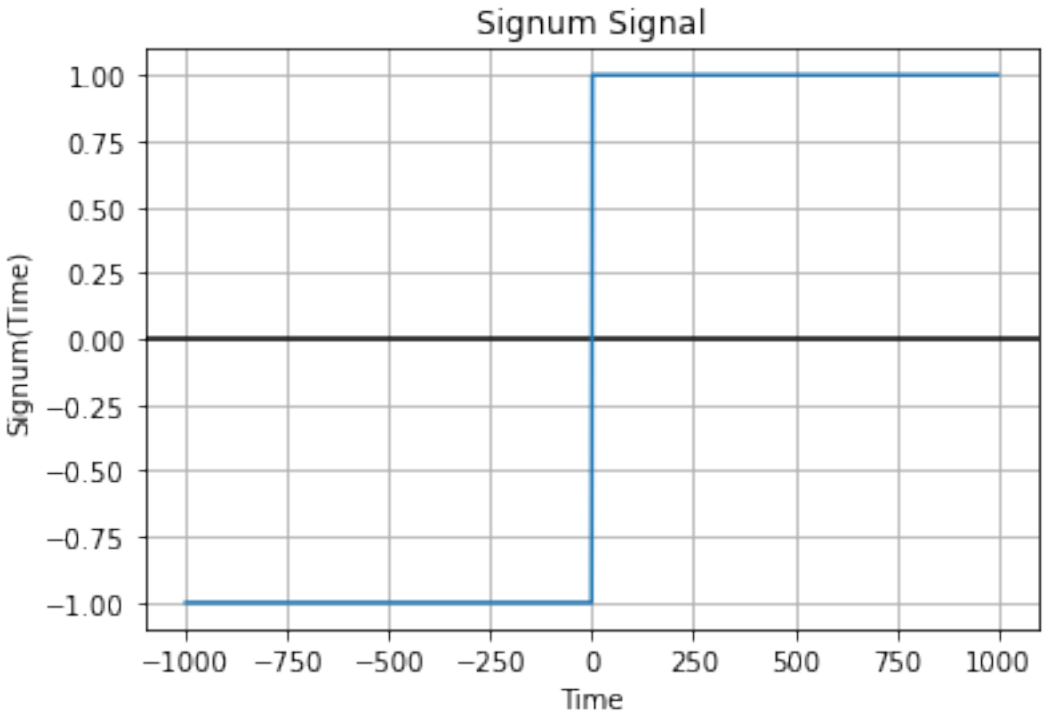



```
#Q1 (b)   Even Signal - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,2*np.pi,0.1)
y = np.cos(2*np.pi*x)
plt.stem(x,y,use_line_collection='true' )
plt.title('Discrete y = cos(t)')
plt.xlabel('t')
plt.ylabel('Amplitude)')
plt.axhline(y=0,color='k')
plt.show()
```



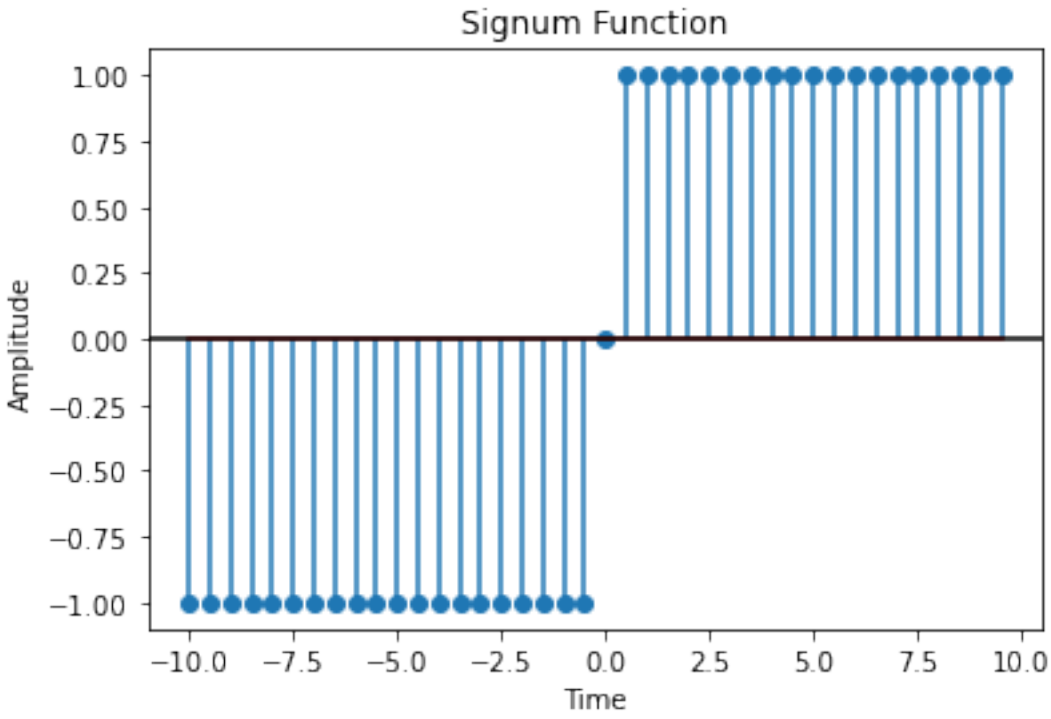

```
#Q1 (c)    Signum Signal

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x=np.arange(-1000,1000)
y=[]
for i in x:
  if (i<0):
    y.append(-1)
  elif(i==0):
    y.append(0)
  elif (i>0):
    y.append(1)

plt.title('Signum Signal')            #this line gives the plot a
title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Signum(Time)')          #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph
```

```
plt.show()                          #this line shows the graph and
successfully terminates the program
```



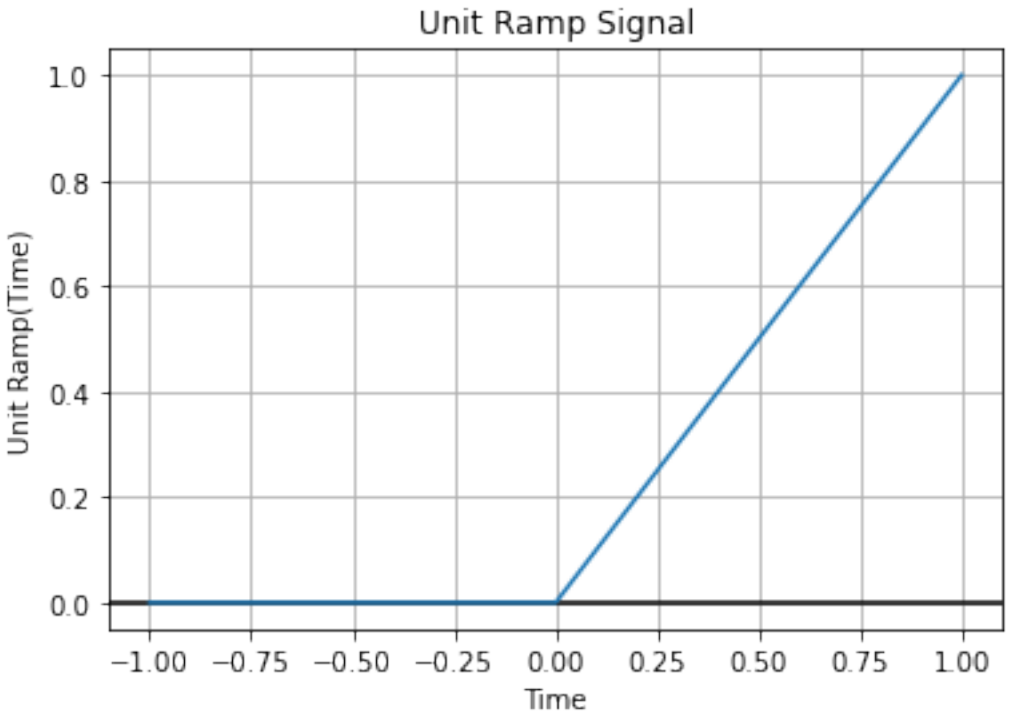

```
#Q1 (c)     Signum Signal - DISCRETE

#defining time for discrete signum signal
x=np.arange(-10,10,0.5)
y=[]
for i in x:
  if (i<0):
    y.append(-1)
  elif(i==0):
    y.append(0)
  elif (i>0):
    y.append(1)


plt.stem(x,y,use_line_collection="True")

#add a horizontal line across the axis.
plt.axhline(y=0, color="black")

# Give title, x, y axis label
plt.title("Signum Function")
plt.xlabel("Time")
plt.ylabel("Amplitude")
```

```
#display the graph
plt.show()
```

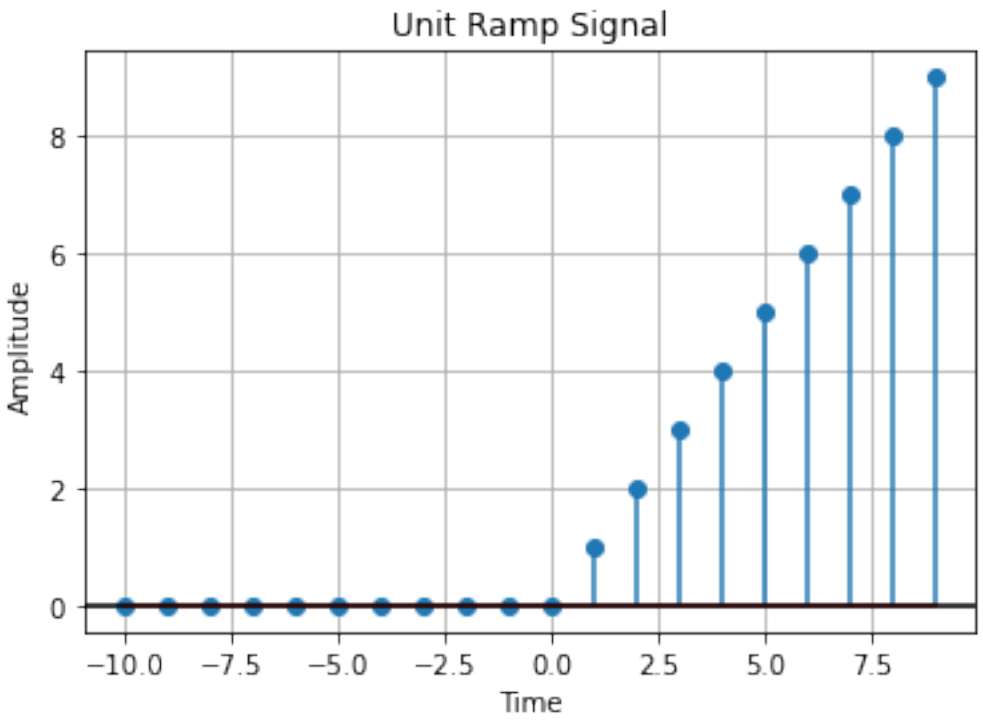



```
#Q1 (d)    Unit Ramp Signal

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x=np.linspace(-1,1,1000,endpoint=True)
y=[]
for i in x:                                  #Code to get Unit Ramp
Signal, by using a for loop
  if (i<=0):
    y.append(0)
  elif (i>0):
    y.append(i)

plt.title('Unit Ramp Signal')            #this line gives the plot a
title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Unit Ramp(Time)')          #this line gives a name for the
y axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
```

```
which adds more detail to the graph
plt.plot(x,y)                        #this line plots the graph

plt.show()                           #this line shows the graph and
successfully terminates the program
```

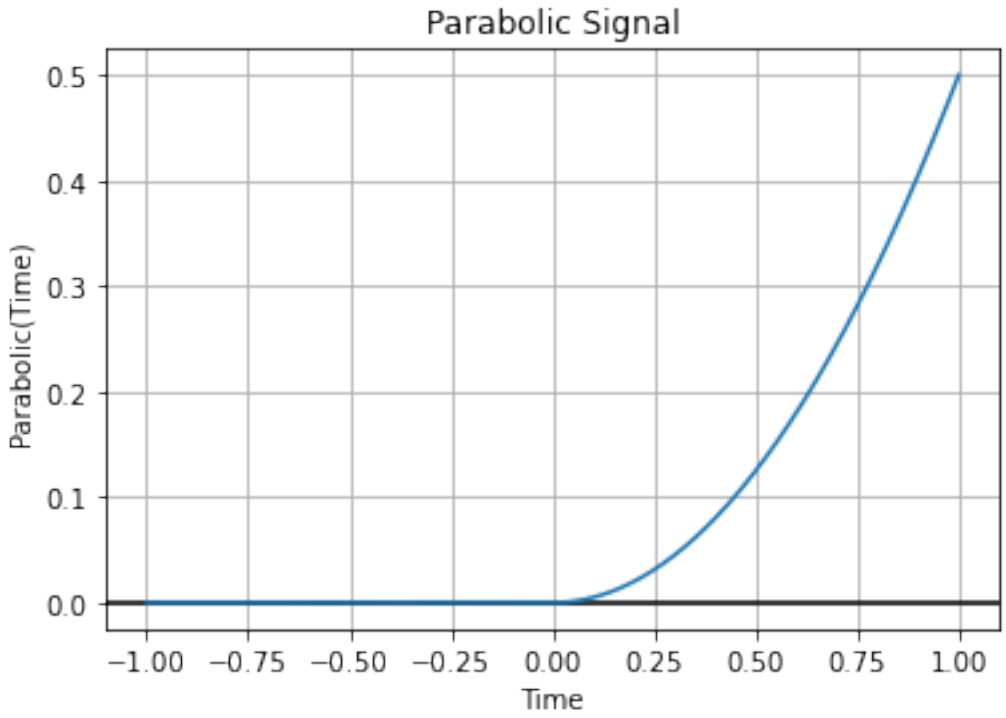



```
#Q1 (d)    Unit Ramp Signal - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:                                  #logic to get the values of
dependant variables for plotting Ramp signal
  if (i<=0):
    y.append(0)
  elif (i>0):
    y.append(i)


plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Unit Ramp Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```

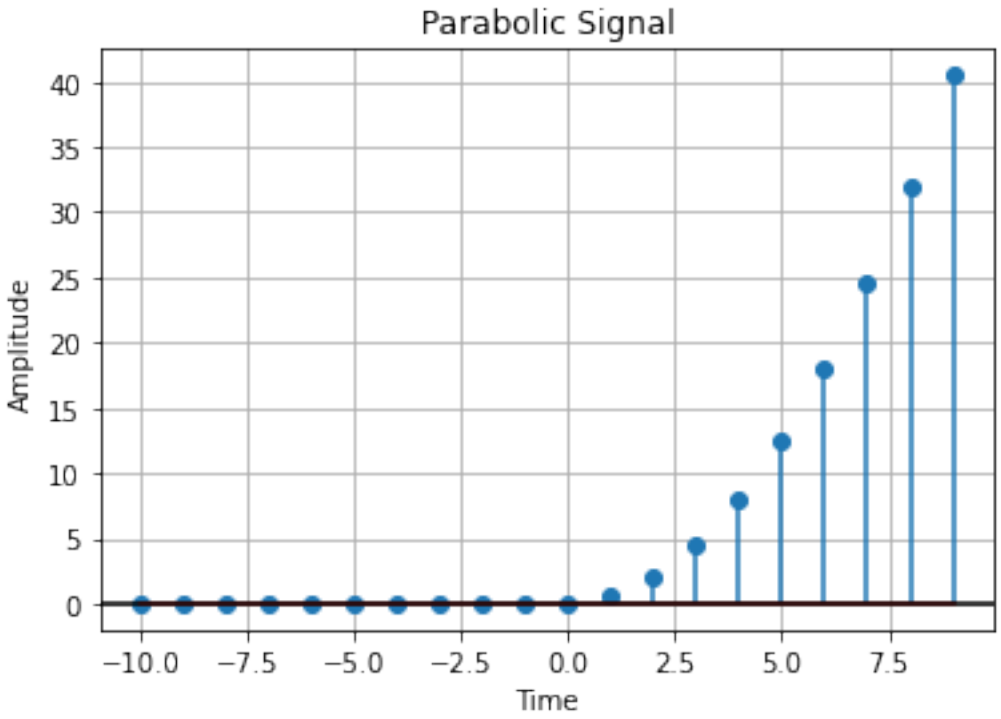



```
#Q1 (e)    Parabolic Signal

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library



x=np.linspace(-1,1,1000,endpoint=True)
y=[]
for i in x:                                  #Code to get parabolic
Signal, by using a for loop
  if (i<=0):
    y.append(0)
  elif (i>0):
    y.append(i*i/2)


plt.title('Parabolic Signal')            #this line gives the plot a
title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Parabolic(Time)')          #this line gives a name for the
y axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
```

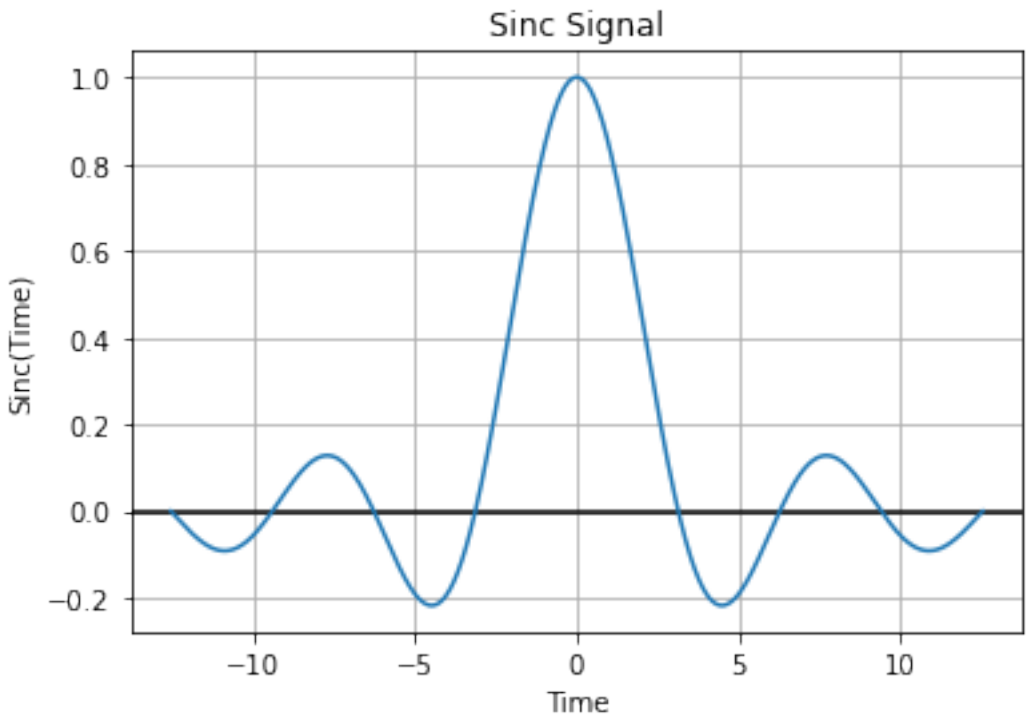
```
which adds more detail to the graph
plt.plot(x,y)                        #this line plots the graph

plt.show()                           #this line shows the graph and
successfully terminates the program
```



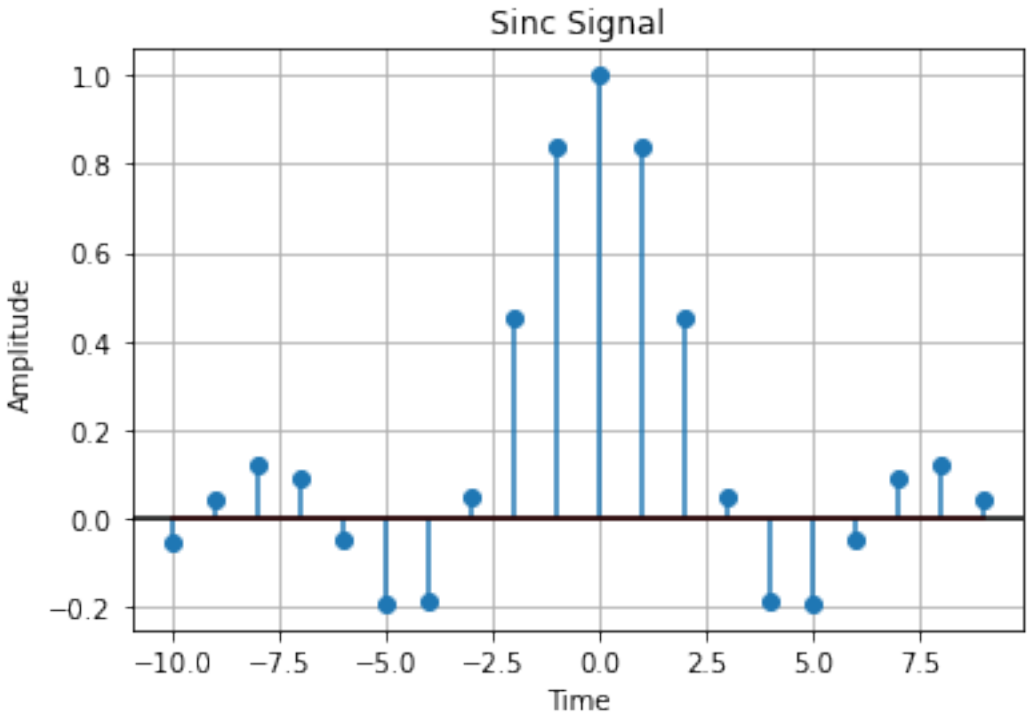

```
#Q1 (d)    Parabolic Signal - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:                                  #logic to get the values of
dependant variables for plotting parabolic signal
  if (i<=0):
    y.append(0)
  elif (i>0):
    y.append(i*i/2)


plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Parabolic Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```




```
#Q1 (f)    Sinc Signal

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x = np.arange(-4*np.pi,4*np.pi,0.001)    #this line makes an array
from -4pi to 4pi, with a step of 0.001
y = (np.sin(x))/x                  #this calculates sin of all values
in our array

plt.title('Sinc Signal')           #this line gives the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Sinc(Time)')          #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph

plt.show()                        #this line shows the graph and
successfully terminates the program
```

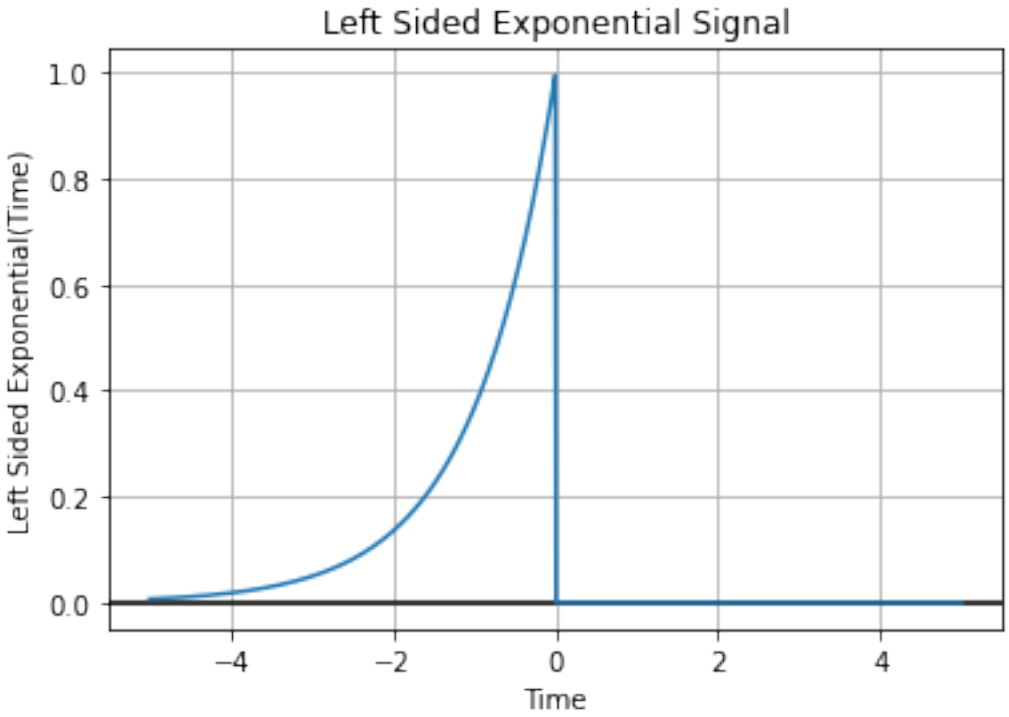



```
#Q1 (f)    Sinc Signal   - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:
    if i == 0:
        y.append(1)
    else:
        y.append((np.sin(i))/i)


plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Sinc Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```

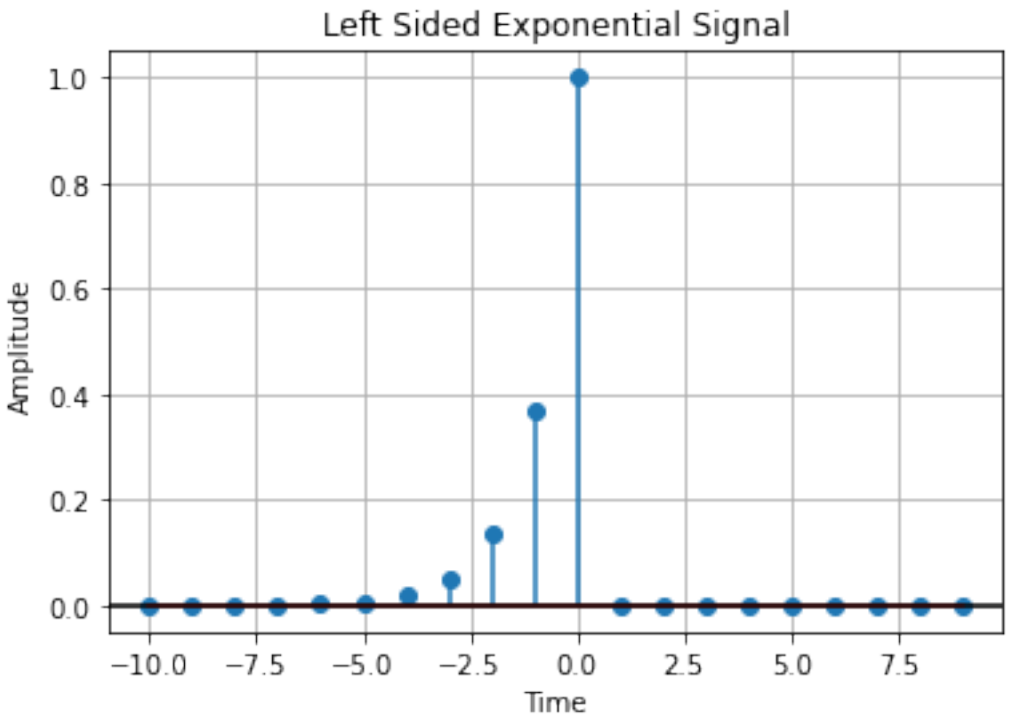



```
#Q1 (g)    Left Sided Exponential Signal


import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x=np.linspace(-5,5,1000)
y=[]
for i in x:
  if (i<=0):                                  #Generating the signal
using an array which uses the exp function
    y.append(np.exp(i))
  elif (i>0):
    y.append(0)

plt.title('Left Sided Exponential Signal')            #this line gives
the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Left Sided Exponential(Time)')          #this line gives a
name for the y axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph
```

```
plt.show()                          #this line shows the graph and
successfully terminates the program
```

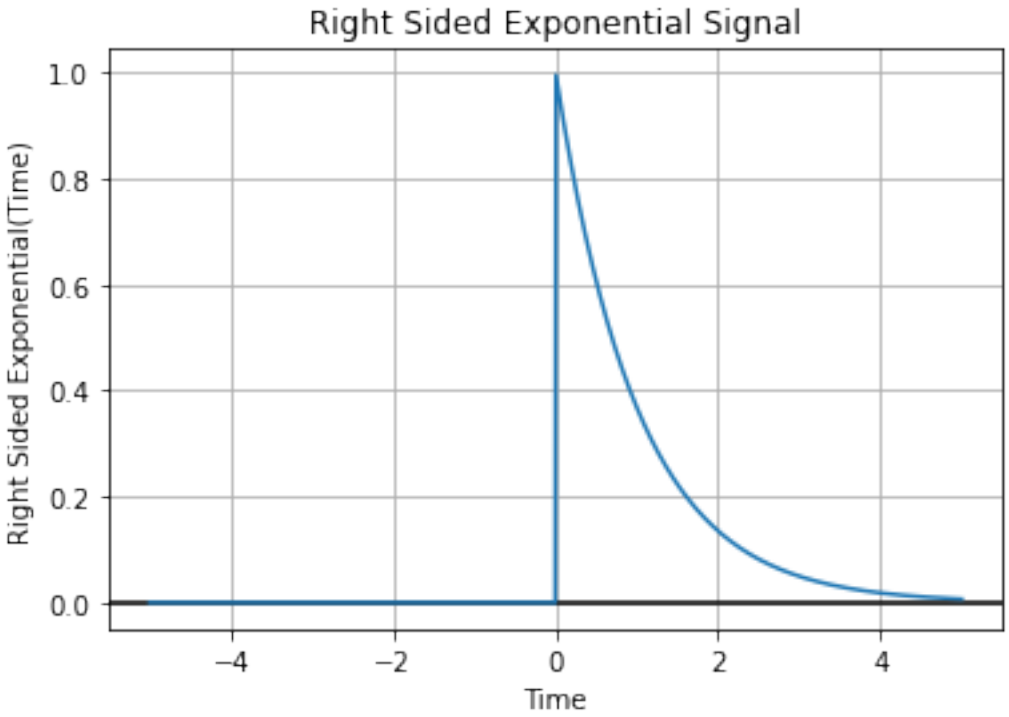



```
#Q1 (g)    Left Sided Exponential Signal - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:
  if (i<=0):
    y.append(np.exp(i))
  elif (i>0):
    y.append(0)


plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Left Sided Exponential Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```

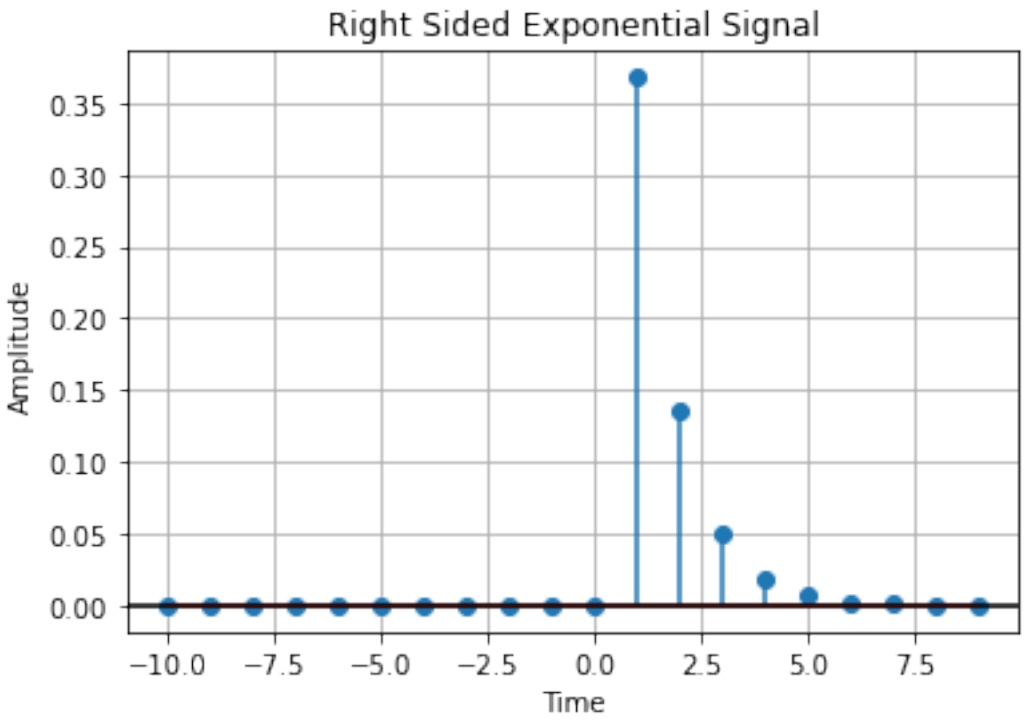



```
#Q1 (g)    Right Sided Exponential Signal


import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x=np.linspace(-5,5,1000)
y=[]
for i in x:
  if (i>0):                                   #Generating the signal
using an array which uses the exp function
    y.append(np.exp(-i))
  elif (i<=0):
    y.append(0)

plt.title('Right Sided Exponential Signal')            #this line
gives the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Right Sided Exponential(Time)')          #this line gives
a name for the y axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph
```

```
plt.show()                          #this line shows the graph and
successfully terminates the program
```

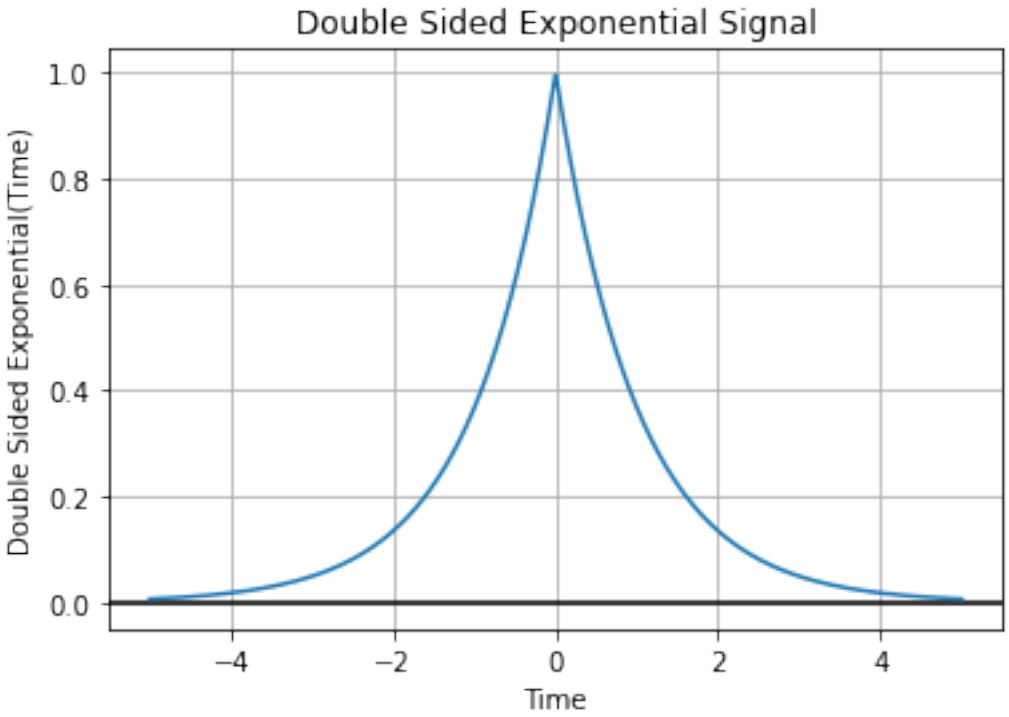



```
#Q1 (g)    Right Sided Exponential Signal - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:
  if (i>0):
    y.append(np.exp(-i))
  elif (i<=0):
    y.append(0)


plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Right Sided Exponential Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```

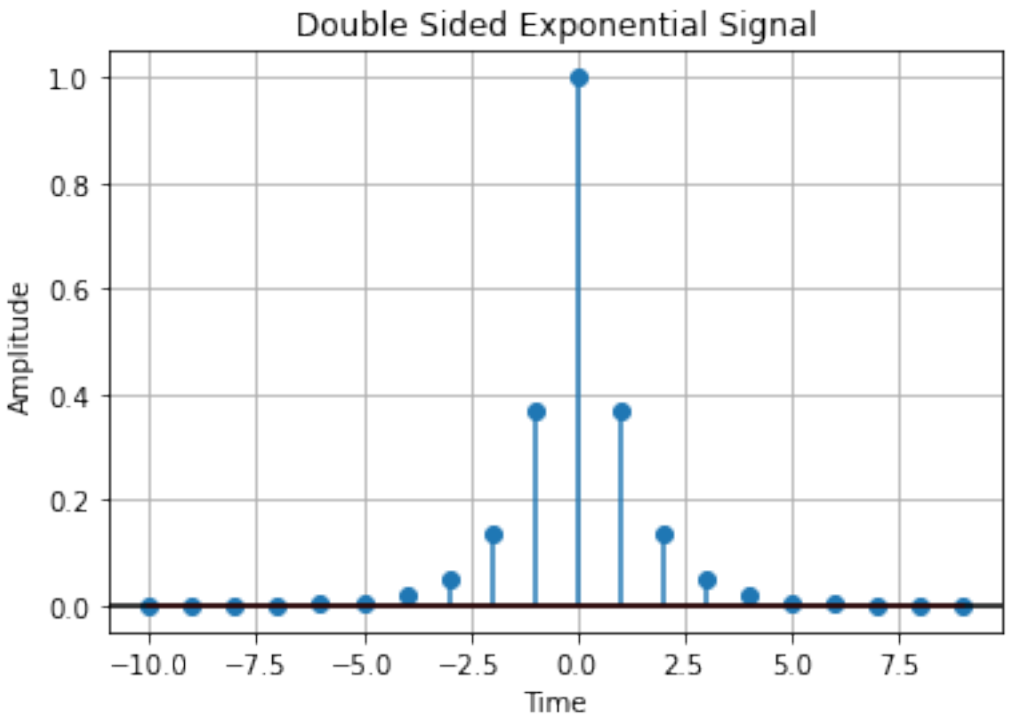



```
#Q1 (g)    Double Sided Exponential Signal


import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x=np.linspace(-5,5,1000)
y=[]
for i in x:
  if (i<=0):                                  #Generating the signal
using an array which uses the exp function
    y.append(np.exp(i))
  elif (i>0):
    y.append(np.exp(-i))

plt.title('Double Sided Exponential Signal')            #this line
gives the plot a title
plt.xlabel('Time')               #this line gives a name for the x
axis and its variables
plt.ylabel('Double Sided Exponential(Time)')          #this line gives
a name for the y axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph
```

```
plt.show()                          #this line shows the graph and
successfully terminates the program
```




```
#Q1 (g)    Double Sided Exponential Signal - DISCRETE

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:
  if (i<=0):                                \
    y.append(np.exp(i))
  elif (i>0):
    y.append(np.exp(-i))


plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Double Sided Exponential Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```


Double Sided Exponential Signal
Amplitude
Time
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5