

① NAME - SANTOSH (CB.EN. V4CLC 2003)

DEPARTMENT - COMPUTER AND COMMUNICATION ENGINEERING (CCE) A

LAB TITLE AND CODE : SIGNAL PROCESSING LAB IACLC 281

EXPERIMENT NO: 2

DATE : 21/09/2021

GENERATION OF ELEMENTARY SIGNALS

* AIM:

Plot continuous-time and discrete-time sequence for the generation of sinusoidal, exponential, impulse, unit step and ramp functions using Python code.

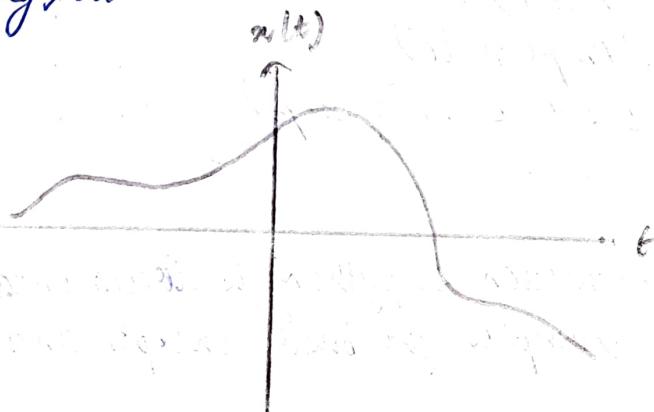
* SOFTWARE REQUIRED :

Synder IDE (Anaconda 3) - Python 3.9.7 (64-bit)

* THEORY :

① Continuous-Time Signal - It is a varying quantity (a signal) whose domain, which is often time, is a continuum (e.g., a connected interval of the reals). That is, the function's domain is an uncountable set. The function itself need not be continuous.

Defined for all time instants and not at specific time instants. The electrical signals derived in proportion with the physical quantities such as temperature, pressure, sound etc. are generally continuous. Applications of these signals include Public Switched Telephone Network (PSTN), FM radio broadcasting, Image Processing, etc.

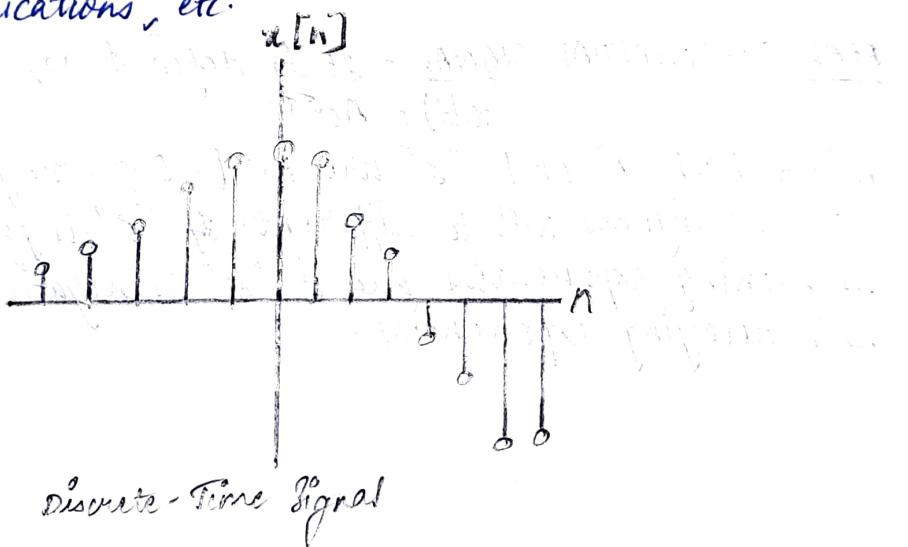


continuous-time signal

②

- ② Discrete-Time Signal - It is a time series consisting of a sequence of a sequence of quantities. Discrete-time views values of variables as occurring at distinct, separate "points in time", or equivalently as being unchanged throughout each non-zero region of time ("time period") - that is, time is viewed as a discrete variable.

Unlike a continuous-time signal, a discrete-time signal is not a function of a continuous argument; however, it may have been obtained by sampling from a continuous-time signal. For example, if you were monitoring the temperature of a room, you would be able to take a measured value of temperature at any time. Applications of these signals include speech processing, SONAR, communications, etc.

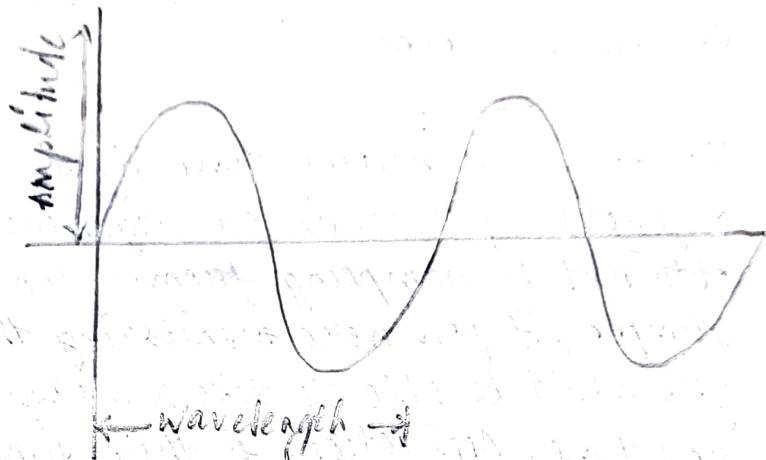


- ③ Sinusoidal signals - These are periodic functions that are based on the sine or cosine function from trigonometry. When the sine wave starts from zero and covers positive values, reaches zero; and again covers negative values, reaches zero, it is said to have completed one cycle or single cycle.

The continuous-time version of a sinusoidal signal, in its most general form, may be written as $x(t) = A \cos(\omega t + \phi)$ where A is the amplitude, ω is the frequency in radians per second, and ϕ is the angle in radians.

(3)

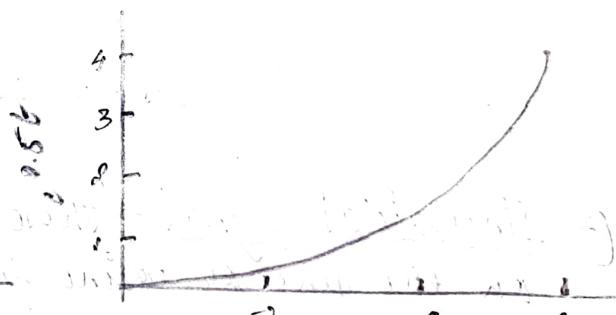
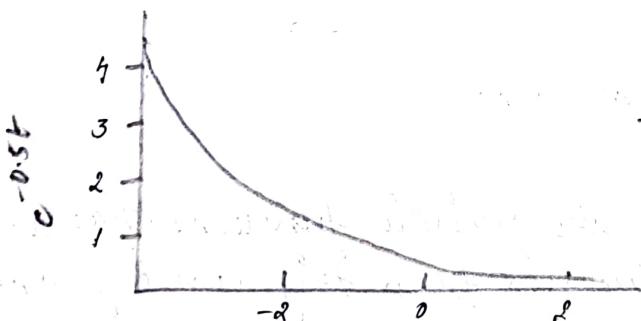
The discrete-time version of a sinusoidal signal, in its most general form, may be written as $a[n] = A \cos(\omega n + \phi)$, where the angular frequency ω must be rational multiple of 2π for the signal to be periodic. Here, the period is measured in samples.



④ REAL EXPONENTIAL SIGNAL - It is defined as,

$$x(t) = Ae^{\delta t}$$

where both "A" and " δ " are real. Depending on the value of the " δ " the signals will be different. If " δ " is positive, the signal $x(t)$ is growing exponential and if " δ " is negative, then the signal $x(t)$ is a decaying exponential.



(a) Decaying Exponential ($\delta < 0$) (b) Growing Exponential ($\delta > 0$)

⑤ COMPLEX EXPONENTIAL SIGNAL - It is defined as,

$$x(t) = Ae^{st}$$

where "s" is a complex variable and it is defined as,

$$s = \delta + j\omega$$

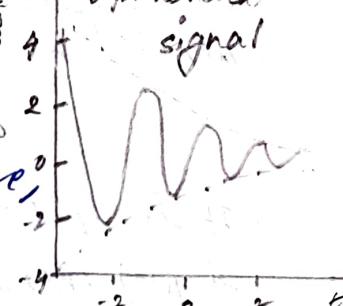
④

Therefore, $x(t) = e^{st} = e^{(s+j\omega)t} = e^{st}e^{j\omega t} - 0$
 Using Euler's identity,

$$e^{j\omega t} = e^{j\theta} = \cos\theta + j\sin\theta - 0$$

Substituting equation ② in equation ①, we have,
 $x(t) = e^{st} [\cos\theta + j\sin\theta]$

Decaying complex exponential signal



⑤

CONTINUOUS-TIME AND DISCRETE-TIME COMPLEX EXPONENTIALS - In both cases, the complex exponential can be expressed through Euler's relation in the form of a real and an imaginary part, both of which are sinusoidal with a phase difference of $\pi/2$ and with an envelope that is a real exponential. When the magnitude of the complex exponential is a constant, then the real and imaginary parts neither grow nor decay with time; in other words, they are purely sinusoidal. In this case for continuous-time, the complex exponential is periodic. For discrete-time, the complex exponential may or may not be periodic depending on whether the sinusoidal real and imaginary components are periodic.

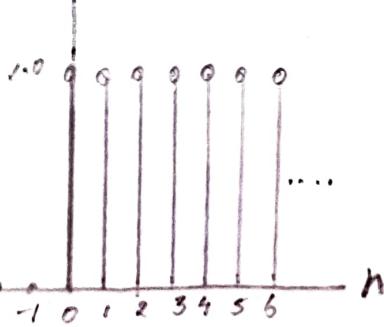
⑥

UNIT STEP FUNCTION - It is the value of which is zero for negative arguments and one for positive arguments. It is an example of the general class of step functions, all of which can be represented as linear combinations of translation of this one.

The discrete-time version of the step function, commonly denoted by $u[n]$, is defined by-

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

Discrete-time version
of step function of
unit amplitude

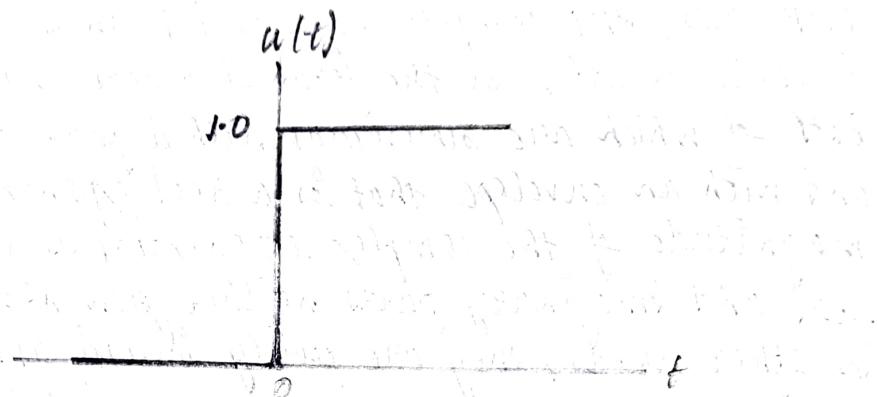


⑤

The continuous-time version of the step function, commonly denoted by $u(t)$, is defined by -

$$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

It is said to exhibit a discontinuity at $t=0$, since the value of $u(t)$ changes instantaneously from 0 to 1 when $t=0$.



continuous-time version of step function of unit impulse

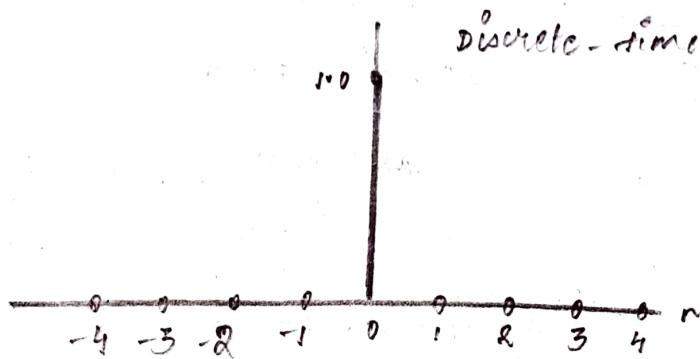
⑥ UNIT IMPULSE FUNCTION - It is a generalized function or distribution over the real numbers, whose value is zero everywhere except at zero, and whose integral over the entire real line is equal to one. It is used as a tool for the normalization of state vectors and also, has uses in probability theory and signal processing.

The discrete-time version of the impulse, commonly denoted by $\delta[n]$, is defined by -

$$\delta[n] = \begin{cases} 1, & n=0 \\ 0, & n \neq 0 \end{cases}$$

$\delta[n]$

discrete-time form of impulse



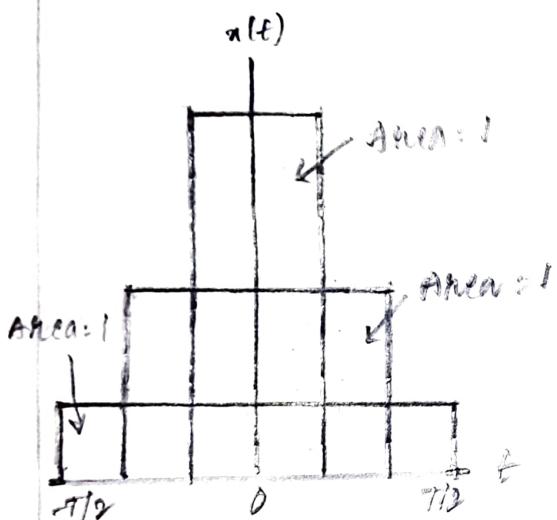
①

The continuous-time version of the unit impulse, commonly denoted by $\delta(t)$, is defined by the following pair of relations -

$$\delta(t) = 0 \text{ for } t \neq 0$$

and

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$



$$a\delta(t)$$

$$a\delta(t)$$

evolution of a rectangular pulse
of unit area into an impulse of
unit strength

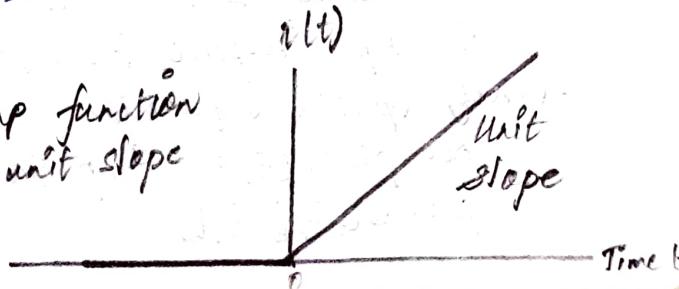
graphical symbol for an
impulse of strength a

② RAMP FUNCTION - It is a unary real function, whose graph is shaped like a ramp. It can be expressed by numerous definitions for example, "0 for negative inputs, output equals input for non-negative inputs." The term "ramp" can also be used for other functions obtained by scaling and shifting, and the function is the unit ramp function (slope 1, starting at 0).

The ~~continuous~~^{continuous}-time version of the ramp function, commonly denoted by $r(t)$, is defined by -

$$r(t) = \begin{cases} t, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

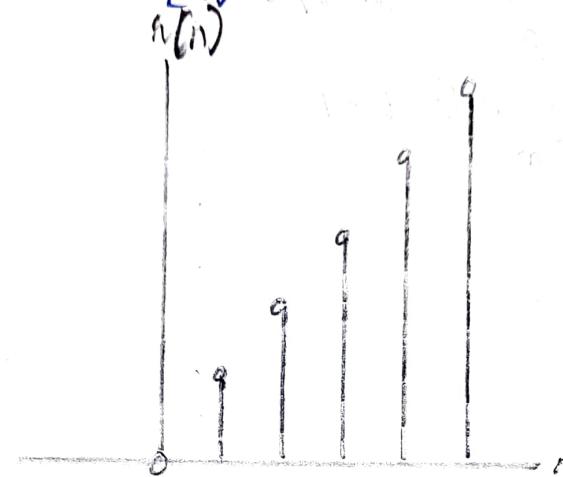
ramp function
of unit slope



⑦

The discrete-time version of the Impulse, commonly denoted by $\delta[n]$, is defined by

$$\delta[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$



Discrete-time version of the impulse function

* GRAPH PLOTTING ALGORITHM:

The following steps are followed:

- ① Define the x-axis and corresponding y-axis values as lists.
- ② Plot them on canvas using `plot()` function.
- ③ Give a name to x-axis and y-axis using `xlabel()` and `ylabel()` functions.
- ④ Give a title to your plot using the `title()` function.
- ⑤ Finally, to view your plots we use the `show()` function.

* PROGRAM WITH COMMENTS:

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

```
def sinusoidal_c(): # Sinusoidal Continuous-Time Signal
    t = np.linspace(-0.02, 0.05, 1000) # linspace() creates
    evenly spaced sequences
    plt.plot(t, 325 * np.sin(2 * np.pi * 50 * t))
    plt.xlabel('t')
```

⑧

```
plt.ylabel('x(t)')
```

```
plt.title('Plot of CT signal: x(t) = 325 * sin(100*pi*t)')
```

```
plt.xlim([0.02, 0.05]) # set the x-limits of the current axes
```

```
plt.show() # To view your plot
```

```
def sinusoidal_d(): # Sinusoidal Discrete-Time Signal
```

$n = np.arange(50)$ # get evenly spaced values within a given interval

```
dt = 0.07 / 50 # Fixes Signal Space Resolution
```

```
x = 325 * np.sin(2 * np.pi * 50 * n * dt)
```

```
plt.xlabel('n')
```

```
plt.ylabel('x[n]')
```

```
plt.title('Plot of DT signal: x[n] = 325 * sin(100 * pi * n)')
```

plt.stem(n, x) # stem plot plots vertical lines at each x position covered under the graph from the baseline to y and places a marker there.

```
plt.show() # To view your plot.
```

```
def exponential_c(): # Exponential Continuous-Time Signals
```

$t = np.linspace(0.02, 0.05, 1000)$ # linspace() creates evenly spaced sequences

plt.subplot(3, 1, 1) # subplot (rows, columns, index)
describes the figure layout

plt.plot(t, np.exp(2j * np.pi * 50 * t).real) # Return the real part of the complex argument

```
plt.xlabel('t')
```

```
plt.ylabel('Re x(t)')
```

```
plt.title('Real part of x(t) = exp(j * 100 * pi * t)')
```

plt.xlim([0.02, 0.05]) # set the x-limits of the current axes

plt.subplot(3, 1, 3) # subplot (rows, columns, index)
describes the figure layout

plt.plot(t, np.exp(2j * np.pi * 50 * t).imag) # Return the imaginary part of the complex argument

⑨

```

plt.xlabel('t')
plt.ylabel('Im z(t)')
plt.title('Imaginary part of z(t) = exp(j * 100 * pi * t)')
plt.xlim([-0.02, 0.05]) # Set the x-limits of the current axis
plt.show() # To view your plot

def exponential_d(): # Exponential Discrete-Time Signal
    n = np.arange(-2, 5) # Get evenly spaced values within a given interval
    arr = [] # Null Array Declaration
    for sample in range(len(n)): # Returns the length of the array
        if n[sample] < 0:
            temp = 1 / (2 ** - (n[sample])) # Conversion of negative power to fractions (2^-x = 1/2^x)
        else:
            temp = 2 ** n[sample]
        arr.append(temp) # Adds a single item to the existing list
    plt.xlabel('n')
    plt.ylabel('x[n]')
    plt.title('Plot of DT signal : x[n] = 2^n')
    plt.stem(n, arr) # Stem plot plots vertical lines at each position covered under the graph from the baseline to y, and places a marker there.
    plt.show() # To view your plot

def step_c(): # Unit Step Continuous-Time Signal
    t = np.linspace(0, 10, 1000) # linspace() creates evenly spaced sequences
    arr = [] # Null array Declaration
    for sample in range(len(t)): # Returns the length of array
        if t[sample] >= 0:
            temp = 1
        else:
            temp = 0
        arr.append(temp)
    plt.xlabel('t')
    plt.ylabel('x(t)')
    plt.title('Plot of CT signal : x(t) = 1 for t >= 0 and 0 for t < 0')
    plt.plot(t, arr)
    plt.show() # To view your plot

```

(10)

```
else
    temp = 0
arr.append(temp) # adds a single item to the existing list
plt.xlabel('t')
plt.ylabel('arr')
plt.title('Unit Step Continuous - Time Function')
plt.step(t, arr) # line forming a series of steps between
data points
plt.show() # To view your plot
```

```
def step_d(): # Unit Step discrete - Time Signal
    n = np.arange(-2, 10) # Get evenly spaced values within
    a given interval
    arr = [] # Null Array Declaration
    for sample in range(len(n)): # Returns the length of array
        if n[sample] >= 0:
            temp = 1
        else:
            temp = 0
        arr.append(temp) # adds a single item to the
existing list
    plt.xlabel('n')
    plt.ylabel('arr[n]')
    plt.title('Unit Step Discrete - Time Function for DSNsa')
    plt.stem(n, arr) # stem plot plots vertical lines at each
    n position under the graph from the baseline to y, and places
    a marker there.
    plt.show() # To view your plot
```

```
def impulse_c(): # Unit Impulse Continuous - Time Signal
    t = np.arange(-2, 10) # Get evenly spaced values within a given
    interval
    arr = [] # Null Array Declaration
```

⑪

```
for sample in range (len(t)): # Returns the length of array
    if t[sample] >= 0:
        temp = 1
    else:
        temp = 0
    arr.append(temp) # Adds a single item to the existing list
plt.xlabel('t')
plt.ylabel('x(t)')
plt.title('Unit Impulse Continuous-Time Function.')
plt.plot(t, arr) # Line forming a series of steps within a
given interval
plt.show() # To view your plot
```

```
def impulse_d(): # Unit Impulse Discrete-Time signal
    n = np.arange(0, 10) # Get evenly spaced values within a
given interval
    arr = [] # Null array declaration
    for sample in range (len(n)): # Returns the length of array
        if n[sample] >= 0:
            temp = 1
        else:
            temp = 0
        arr.append(temp) # Adds a single item to the existing list
    plt.xlabel('n')
    plt.ylabel('x(n)')
    plt.title('Unit Impulse Discrete-Time Function for 0 ≤ n ≤ 9')
    plt.stem(n, arr) # stem plot plots vertical lines at each
position covered under the graph from the baseline to y. and
places a marker there.
    plt.show() # To view your plot
```

10

```

def ramp_c(t): # Ramp Function Continuous-Time Signal
    t = np.linspace(0, 10, 1000) # linspace() creates evenly spaced sequences
    plt.plot(t, t)
    plt.xlabel('t')
    plt.ylabel('n(t)')
    plt.title('Ramp Continuous-Time Function')
    plt.xlim([0, 10]) # Set the x-limits of the current axes
    plt.show() # To view your plot

```

```

def ramp_d(n): # Ramp Function Discrete-Time Signal
    n = np.arange(0, 10) # linspace() creates evenly spaced sequences
    plt.xlabel('n')
    plt.ylabel('n(t)')
    plt.title('Ramp Discrete-Time Function for 0 ≤ n ≤ 9')
    plt.stem(n, n) # stem plot plots vertical lines at each n position covered under the graph from the baseline to y, and places a marker there
    plt.show() # To view your plot

```

Main

```

while True: # This simulates a do-loop
    choice = input("MENU: In 1. Sinusoidal Continuous-Time Signal. In 2. Sinusoidal Discrete-Time Signal. In 3. Exponential Continuous-Time Signal. In 4. Exponential Discrete-Time Signal. In 5. Unit Step Continuous-Time Signal. In 6. Unit Step Discrete-Step Signal. In 7. Unit Impulse Continuous-Time Signal. In 8. Unit Impulse Discrete-Time Signal. In 9. Ramp Function Continuous-Time Signal. In 10. Ramp Function Discrete-Time Signal. In 11. Exit. Enter the number corresponding to the menu to implement the choice: ") # Menu Driven Implementation

```

(13)

```
if choice == str(1): # str() returns the string version of  
the variable "choice"  
    sinusoidal_c() # Sinusoidal Continuous-Time Signal  
    break  
elif choice == str(2):  
    sinusoidal_d() # Sinusoidal Discrete-Time signal  
    break  
elif choice == str(3):  
    exponential_c() # Exponential continuous-Time signal  
    break  
elif choice == str(4):  
    exponential_d() # Exponential discrete-Time signal  
    break  
elif choice == str(5):  
    step_c() # Unit Step Continuous-Time signal  
    break  
elif choice == str(6):  
    step_d() # Unit Step Discrete-Time signal  
    break  
elif choice == str(7):  
    step_impulse_c() # Unit Impulse Continuous-Time signal  
    break  
elif choice == str(8):  
    impulse_d() # Unit Impulse Discrete-Time signal  
    break  
elif choice == str(9):  
    ramp_c() # Ramp Function Continuous-Time signal  
    break  
elif choice == str(10):  
    ramp_d() # Ramp Function Discrete-Time signal  
    break  
elif choice == str(11):  
    break # Exit loop
```

(14)

else :

print ("Error : Invalid Input! Please try again. \n")

+ INFEREN CE:

Viscaalize the complex exponential signal and real sinusoids
and demonstrate other simple elementary signals and
results verified.

Sinusoidal Continuous-Time Signal

Spyder (Python 3.8)

```
# -*- coding: utf-8 -*-
...
Created on Tue Sep 21 13:56:03 2021
@author: Dell
...
import numpy as np
import matplotlib.pyplot as plt

def sinusoidal_c(): # Sinusoidal Continuous-Time Signal
    t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
    plt.plot(t, 325 * np.sin(2 * np.pi * 50 * t))
    plt.xlabel('t')
    plt.ylabel('x(t)')
    plt.title('Plot of CT signal: x(t) = 325*sin(100*pi*t)')
    plt.ylim([-300, 300]) # Set the x-limits of the current axes
    plt.show() # To view your plot

def sinusoidal_d(): # Sinusoidal Discrete-Time Signal
    n = np.arange(50) # Get evenly spaced values within a given interval
    dt = 0.001 # Fixed Signal Space Resolution
    x = 325 * np.sin(2 * np.pi * 50 * n * dt)
    plt.xlabel('n')
    plt.ylabel('x(n)')
    plt.title('Plot of DT signal: x[n] = 325*sin(100*pi*n)')
    plt.plot(n, x) # Subplots vertical lines at each x position covered under the graph from the baseline
    plt.show() # To view your plot

def exponential_c(): # Exponential Continuous-Time Signal
    t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
    plt.subplot(1,1,1) # subplot(rows, columns, index) describes the figure layout
    plt.plot(t, np.exp(2j * np.pi * 50 * t).real) # Return the real part of the complex argument
    plt.xlabel('t')
    plt.ylabel('x(t)')
    plt.title('Plot of CT signal: x(t)=exp(j*100*pi*t)')
    plt.ylim([-300, 300]) # Set the x-limits of the current axes
    plt.show() # To view your plot

def exponential_d(): # Exponential Discrete-Time Signal
    n = np.arange(50) # Get evenly spaced values within a given interval
    dt = 0.001 # Fixed Signal Space Resolution
    x = np.exp(2j * np.pi * 50 * n * dt).real # Return the real part of the complex argument
    plt.subplot(1,1,1) # subplot(rows, columns, index) describes the figure layout
    plt.plot(n, x) # Subplots vertical lines at each x position covered under the graph from the baseline
    plt.show() # To view your plot

In [1]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

```

Variable explorer Help File

Console 1/A

In [1]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

MENU:

- 1. Sinusoidal Continuous-Time Signal.
- 2. Sinusoidal Discrete-Time Signal.
- 3. Exponential Continuous-Time Signal.
- 4. Exponential Discrete-Time Signal.
- 5. Unit Step Continuous-Time Signal.
- 6. Unit Step Discrete-Time Signal.
- 7. Unit Impulse Continuous-Time Signal.
- 8. Unit Impulse Discrete-Time Signal.
- 9. Ramp Function Continuous-Time Signal.
- 10. Ramp Function Discrete-Time Signal.
- 11. Exit

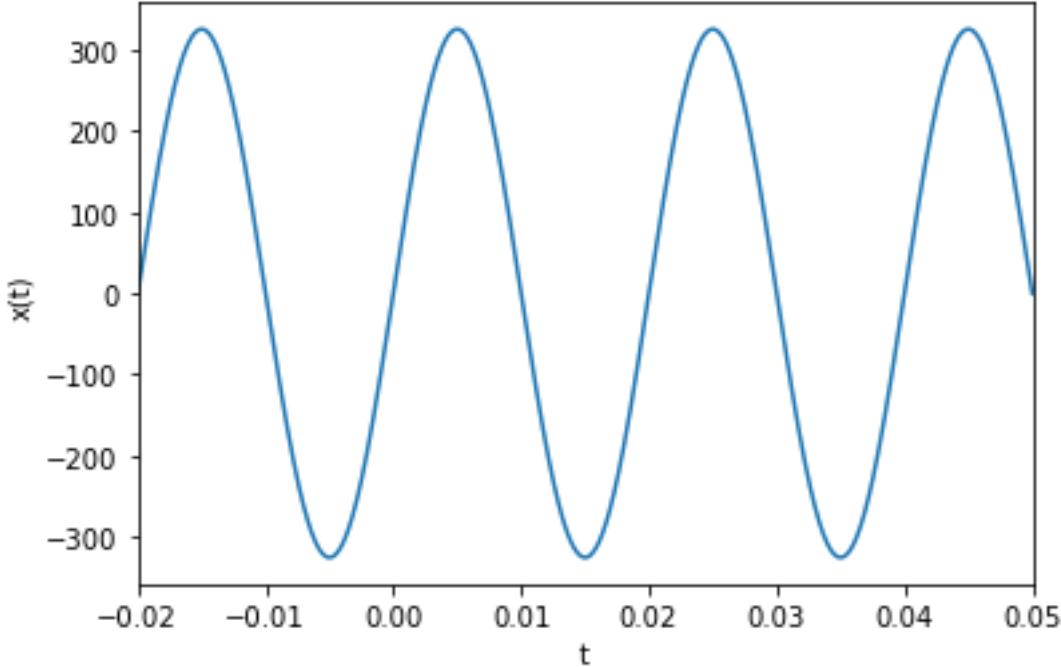
Enter the number corresponding to the menu to implement the choice: 1

In [2]:

LSP Python: ready conda (Python 3.8.8) Line 112, Col 14 UTF-8 CRLF RW Mem 86%

26°C ↻ ⌂ ENG 0047 28-09-2021

Plot of CT signal: $x(t) = 325\sin(100\pi t)$



Sinusoidal Discrete-Time Signal

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

E:\Plan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 2\Experiment 2.py

Experiment 2.py

```

1 # -*- coding: utf-8 -*-
...
3 Created on Tue Sep 21 13:56:03 2021
4
5 @author: Dell
6
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 def sinusoidal_c(): # Sinusoidal Continuous-Time Signal
12     t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
13     dt = 0.0001 # Fix the Signal Space Resolution
14     x = 325 * np.sin(2 * np.pi * 50 * t)
15     plt.xlabel('t')
16     plt.ylabel('x(t)')
17     plt.title('Plot of CT signal: x(t) = 325*sin(100*pi*t)')
18     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
19     plt.show() # To view your plot
20
21 def sinusoidal_d(): # Sinusoidal Discrete-Time Signal
22     n = np.arange(50) # Get evenly spaced values within a given interval
23     dt = 0.0001 # Fix the Signal Space Resolution
24     x = 325 * np.sin(2 * np.pi * 50 * n * dt)
25     plt.xlabel('n')
26     plt.ylabel('x(n)')
27     plt.title('Plot of DT signal: x[n] = 325*sin(100*pi*n)')
28     plt.stem(n, x) # stem() plots vertical lines at each x position covered under the graph from the baseline up and places a marker there.
29     plt.show() # To view your plot
30
31 def exponential_c(): # Exponential Continuous-Time Signal
32     t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
33     plt.subplot(1,1,1) # subplot(rows, columns, index) describes the figure layout
34     plt.plot(t, np.exp(2j * np.pi * 50 * t).real) # Return the real part of the complex argument
35     plt.xlabel('t')
36     plt.ylabel('x(t)')
37     plt.title('Plot of CT signal: x(t)=exp(j*100*pi*t)')
38     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
39     plt.show() # To view your plot
40
41 def exponential_d(): # Exponential Discrete-Time Signal
42     n = np.arange(50) # Get evenly spaced values within a given interval
43     dt = 0.0001 # Fix the Signal Space Resolution
44     x = np.exp(2j * np.pi * 50 * n).real # Return the real part of the complex argument
45     plt.xlabel('n')
46     plt.ylabel('x(n)')
47     plt.title('Plot of DT signal: x[n]=exp(j*100*pi*n)')
48     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
49     plt.show() # To view your plot
50

```

Plot of DT signal: $x[n] = 325 \sin(100\pi n)$

Variable explorer Help File

In [2]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

Console 1/A

MENU:

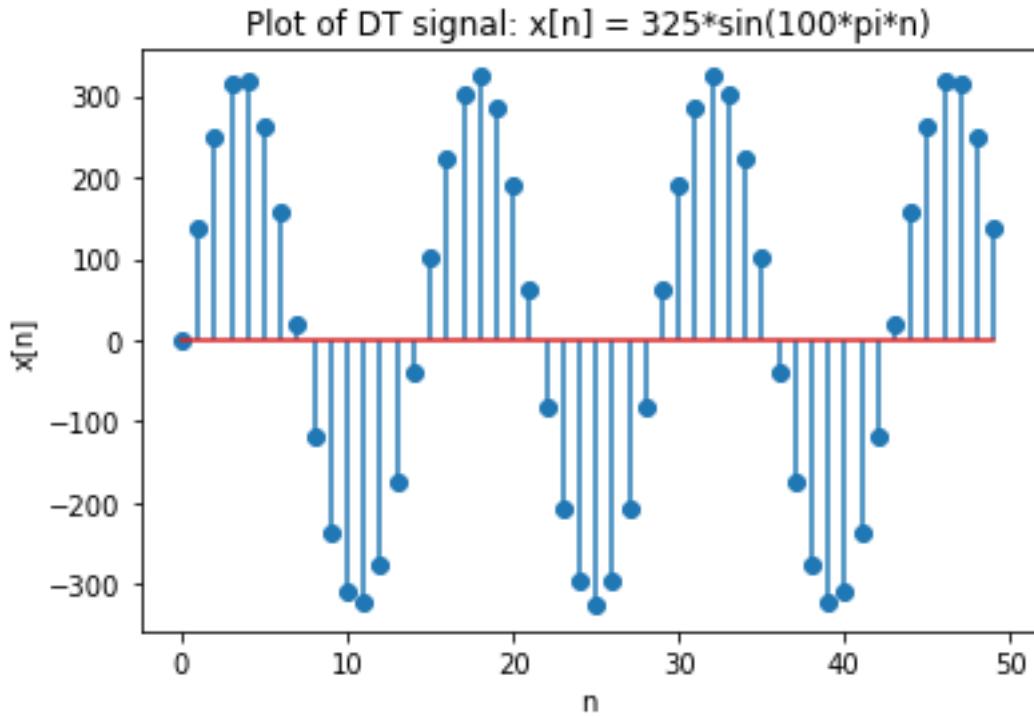
- 1. Sinusoidal Continuous-Time Signal.
- 2. Sinusoidal Discrete-Time Signal.
- 3. Exponential Continuous-Time Signal.
- 4. Exponential Discrete-Time Signal.
- 5. Unit Step Continuous-Time Signal.
- 6. Unit Step Discrete-Time Signal.
- 7. Unit Impulse Continuous-Time Signal.
- 8. Unit Impulse Discrete-Time Signal.
- 9. Ramp Function Continuous-Time Signal.
- 10. Ramp Function Discrete-Time Signal.
- 11. Exit

 Enter the number corresponding to the menu to implement the choice: 2

In [3]:

LSP Python: ready conda (Python 3.8.8) Line 112, Col 14 UTF-8 CRLF RW Mem 84%

26°C 28-09-2021



Exponential Continuous-Time Signal

Spawner (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

E:\Plan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 2\Experiment 2.py

Experiment 2.py

```

1 # -*- coding: utf-8 -*-
...
3 Created on Tue Sep 21 13:56:03 2021
4
5 @author: Dell
6
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 def sinusoidal_c(): # Sinusoidal Continuous-Time Signal
12     t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
13     plt.plot(t, 325 * np.sin(2 * np.pi * 50 * t))
14     plt.xlabel('t')
15     plt.ylabel('x(t)')
16     plt.title('Plot of CT signal: x(t) = 325*sin(100*pi*t)')
17     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
18     plt.show() # To view your plot
19
20 def sinusoidal_d(): # Sinusoidal Discrete-Time Signal
21     n = np.arange(50) # Get evenly spaced values within a given interval
22     dt = 0.001/50 # Fix Signal Space Resolution
23     x = 325 * np.sin(2 * np.pi * 50 * n * dt)
24     plt.xlabel('n')
25     plt.ylabel('x(n)')
26     plt.title('Plot of DT signal: x[n] = 325*sin(100*pi*n)')
27     plt.stem(n, x) # stem() plots vertical lines at each x position covered under the graph from the baseline.
28     plt.show() # To view your plot
29
30 def exponential_c(): # Exponential Continuous-Time Signal
31     t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
32     plt.subplot(1,1,1) # subplot(rows, columns, index) describes the figure layout
33     plt.plot(t, np.exp(2j * np.pi * 50 * t).real) # Return the real part of the complex argument
34     plt.xlabel('t')
35     plt.ylabel('Re x(t)')
36     plt.title('Real part of x(t)=exp(j*100*pi*t)')
37     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
38     plt.subplot(1,1,1) # subplot(rows, columns, index) describes the figure layout
39     plt.plot(t, np.exp(2j * np.pi * 50 * t).imag) # Return the imaginary part of the complex argument
40     plt.xlabel('t')
41     plt.ylabel('Im x(t)')
42     plt.title('Imaginary part of x(t)=exp(j*100*pi*t)')
43     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
44     plt.show() # To view your plot
45

```

Real part of $x(t)=\exp(j*100*\pi*t)$

Imaginary part of $x(t)=\exp(j*100*\pi*t)$

Variable explorer Help File

In [3]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

Console 1/A

MENU:

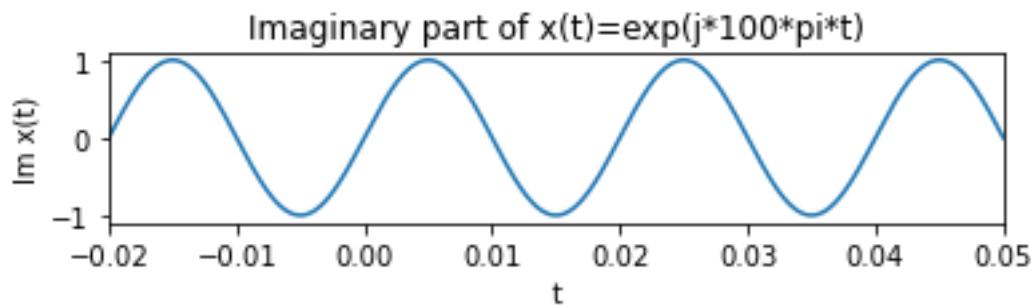
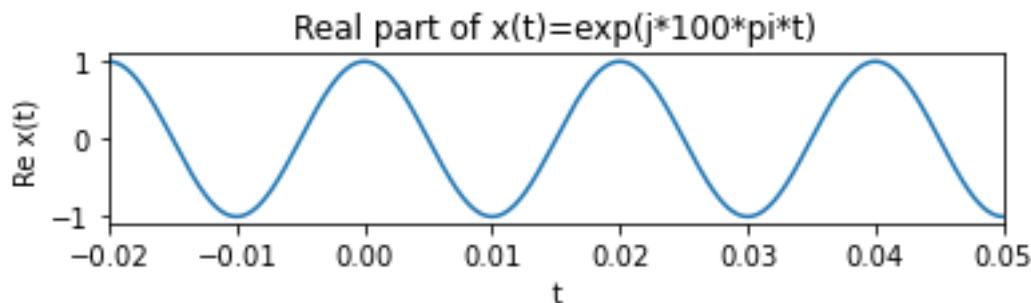
- 1. Sinusoidal Continuous-Time Signal.
- 2. Sinusoidal Discrete-Time Signal.
- 3. Exponential Continuous-Time Signal.
- 4. Exponential Discrete-Time Signal.
- 5. Unit Step Continuous-Time Signal.
- 6. Unit Step Discrete-Time Signal.
- 7. Unit Impulse Continuous-Time Signal.
- 8. Unit Impulse Discrete-Time Signal.
- 9. Ramp Function Continuous-Time Signal.
- 10. Ramp Function Discrete-Time Signal.
- 11. Exit

 Enter the number corresponding to the menu to implement the choice: 3

In [4]:

LSP Python: ready conda (Python 3.8.8) Line 112, Col 14 UTF-8 CRLF RW Mem 84%

26°C 28-09-2021



Exponential Discrete-Time Signal

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Experiment 2.py

```

19
20 def sinusoidal_d(): # Sinusoidal Discrete-Time Signal
21     n = np.arange(50) # Get evenly spaced values within a given interval
22     dt = 0.075/50 # Fixes Signal Space Resolution
23     x = 325 * np.sin(2 * np.pi * 50 * n * dt)
24     plt.stem(n, x)
25     plt.xlabel('n')
26     plt.title('Plot of DT signal: x[n] = 325sin(100*pi*n)')
27     plt.stem(n, x) # Stem plots vertical lines at each x position covered under the graph from the baseline to y_i and places a marker there.
28     plt.show() # to view your plot
29
30 def exponential_c(): # Exponential Continuous-Time Signal
31     t = np.linspace(-0.02, 0.05, 1000) # linspace() creates evenly spaced sequences
32     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
33     plt.plot(t, np.exp(2 * np.pi * 50 * t).real) # Return the real part of the complex argument
34     plt.xlabel('t')
35     plt.ylabel('x(t)')
36     plt.title('Real part of x(t)=exp(j*100*pi*t)')
37     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
38     plt.plot(t, np.exp(2 * np.pi * 50 * t).imag) # Return the imaginary part of the complex argument
39     plt.xlabel('t')
40     plt.ylabel('Im x(t)')
41     plt.title('Imaginary part of x(t)=exp(j*100*pi*t)')
42     plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
43     plt.show() # to view your plot
44
45 def exponential_d(): # Exponential Discrete-Time Signal
46     n=np.arange(-2,5) # Get evenly spaced values within a given interval
47     arr=[] # Empty Array Declaration
48     for sample in range(len(n)): # Returns the length of the array
49         if [sample]<0:
50             temp=1/2**(-(n[sample])) # Conversion of negative powers to fractions (2^-1 - 1/2)
51         else:
52             temp=2**n[sample])
53         arr.append(temp) # Adds a single item to the existing list
54     plt.stem(n, arr)
55     plt.xlabel('n')
56     plt.ylabel('x[n]')
57     plt.title('Plot of DT signal: x[n]=2^n')
58     plt.stem(n, arr) # Stem plots vertical lines at each x position covered under the graph from the baseline to y_i and places a marker there.
59     plt.show() # to view your plot
60
61 def step_c(): # Unit Step Continuous-Time Signal
62     t=np.linspace(-2, 10, 1000) # linspace() creates evenly spaced sequences
63     arr=np.zeros(1000) # Null Array Declaration

```

Plot of DT signal: $x[n]=2^n$

Variable explorer Help File

In [4]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

Console 1/A

MEMO:

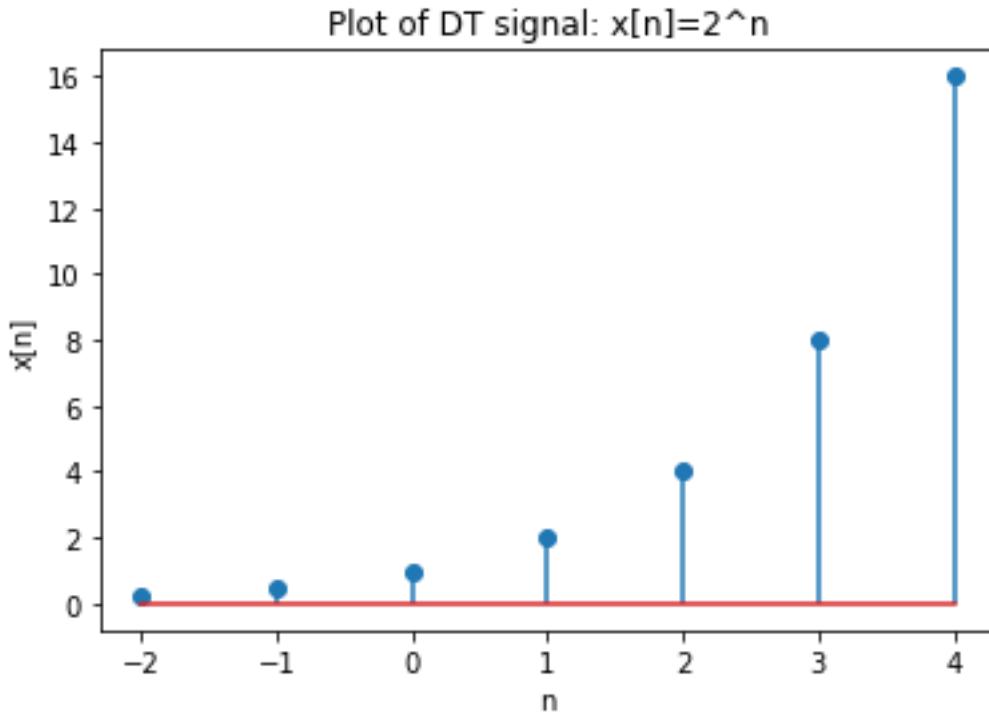
1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 4

In [5]:

LSP Python: ready conda (Python 3.8.8) Line 112, Col 14 UTF-8 CRLF RW Mem 85%

26°C 28-09-2021

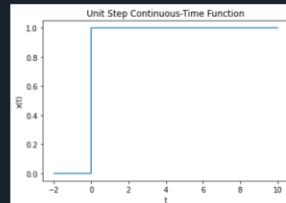


Unit Step Continuous-Time Signal

Spawner (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Plan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 2.py
Experiment 2.py
33 plt.plot(t, np.exp(2j * np.pi * 50 * t).real) # Return the real part of the complex argument
34 plt.xlabel('Re x(t)')
35 plt.title('Real part of x(t)=exp(j*100*pi*t)')
36 plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
37 plt.subplot(1, 2, 1) # subplot(rows, columns, index) describes the figure layout
38 plt.plot(t, np.exp(2j * np.pi * 50 * t).imag) # Return the imaginary part of the complex argument
39 plt.xlabel('Im x(t)')
40 plt.title('Imaginary part of x(t)=exp(j*100*pi*t)')
41 plt.ylim([-0.02, 0.05]) # Set the x-limits of the current axes
42 plt.show() # To view your plot
43
44 exponential_d(): # Exponential Discrete-Time Signal
45 n=np.arange(-2,2) # Get evenly spaced values within a given interval
46 arr=[1] # Initial Array Declaration
47 for sample in range(len(n)): # Returns the length of the array
48     if n[sample]<0:
49         temp=1/(2**(-n[sample])) # Conversion of negative power to fractions (2^-1 = 1/2)
50     else:
51         temp=(2**n[sample])
52     arr.append(temp) # Adds a single item to the existing list
53 plt.xlabel('n')
54 plt.ylabel('x(n)')
55 plt.title('Plot of DT signal: x[n]=2^n')
56 plt.step(n,arr) # Step plot plots vertical lines at each x position covered under the graph from the baseline up and places a marker there.
57 plt.show() # To view your plot
58
59 def step_c(): # Unit Step Continuous-Time Signal
60 t=np.linspace(-2, 10, 1000) # linspace() creates evenly spaced sequences
61 arr=[1] # Initial Array Declaration
62 for sample in range(len(t)): # Returns the length of the array
63     if t[sample]<0:
64         temp=0
65     else:
66         temp=1
67     arr.append(temp) # Adds a single item to the existing list
68 plt.xlabel('t')
69 plt.ylabel('x(t)')
70 plt.title('Unit Step Continuous-Time Function')
71 plt.step(t,arr) # Line forming a series of steps between data points
72 plt.show() # To view your plot
73
74 def step_d(): # Unit Step Discrete-Time Signal
75 n=np.arange(-2, 10) # Get evenly spaced values within a given interval
76 arr=[1] # Initial Array Declaration
77
```

Unit Step Continuous-Time Function



Variable explorer Help File

In [5]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

Console 1/A

MEMO:

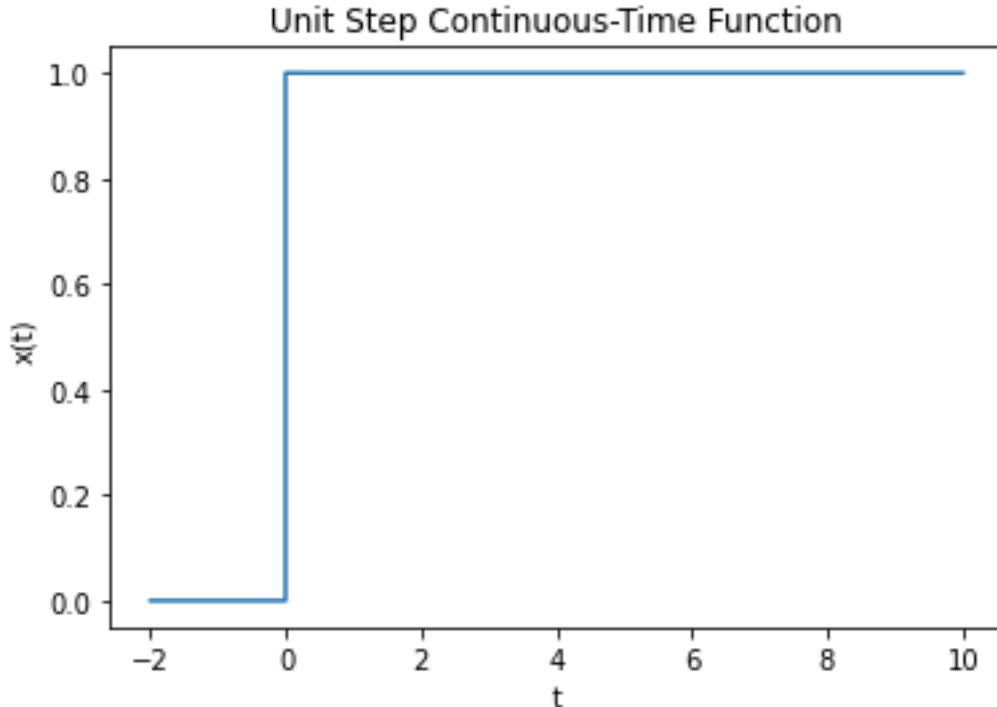
- 1. Sinusoidal Continuous-Time Signal.
- 2. Sinusoidal Discrete-Time Signal.
- 3. Exponential Continuous-Time Signal.
- 4. Exponential Discrete-Time Signal.
- 5. Unit Step Continuous-Time Signal.
- 6. Unit Step Discrete-Time Signal.
- 7. Unit Impulse Continuous-Time Signal.
- 8. Unit Impulse Discrete-Time Signal.
- 9. Ramp Function Continuous-Time Signal.
- 10. Ramp Function Discrete-Time Signal.
- 11. Exit

Enter the number corresponding to the menu to implement the choice: 5

In [6]:

LSP Python: ready conda (Python 3.8.8) Line 112, Col 14 UTF-8 CRLF RW Mem 85%

26°C 28-09-2021

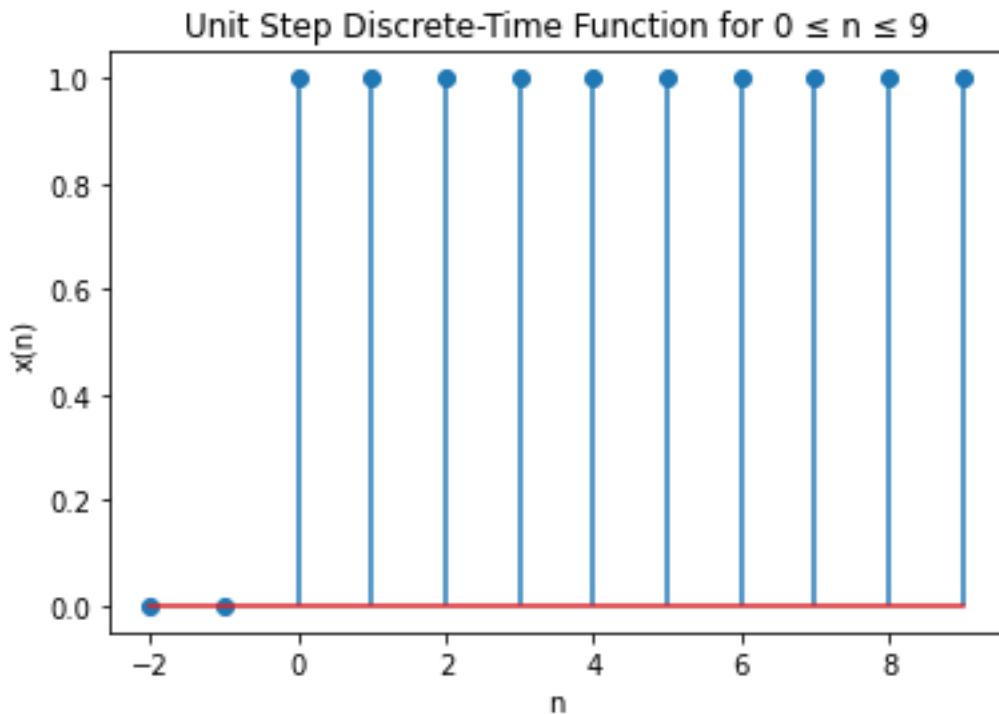


Unit Step Discrete-Time Signal

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Plan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 2.py
Experiment 2.py
48 arr=[ ] # Null Array Declaration
49 for sample in range(len(n)): # Returns the length of the array
50     if n[sample]<0:
51         temp=1/(2**(-(n[sample]))) # Conversion of negative power to fractions (2^-1 = 1/2)
52     else:
53         temp=(2**(n[sample]))
54     arr.append(temp) # Adds a single item to the existing list
55 plt.xlabel('n')
56 plt.ylabel('x(n)')
57 plt.title('Plot of DT signal: x[n]=2^n')
58 plt.step(t, arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
59 plt.show() # To view your plot
60
61 def step_c(): # Unit Step Continuous-Time Signal
62     t=np.linspace(-2, 10, 1000) # linspace() creates evenly spaced sequences
63     arr=[ ] # Null Array Declaration
64     for sample in range(len(t)): # Returns the length of the array
65         if n[sample]>0:
66             temp=1
67         else:
68             temp=0
69         arr.append(temp) # Adds a single item to the existing list
70     plt.xlabel('t')
71     plt.ylabel('x(t)')
72     plt.title('Unit Step Continuous-Time Function')
73     plt.step(t, arr) # Line forming a series of steps between data points
74     plt.show() # To view your plot
75
76 def step_d(): # Unit Step Discrete-Time Signal
77     n=np.arange(-10) # Get evenly spaced values within a given interval
78     arr=[ ] # Null Array Declaration
79     for sample in range(len(n)): # Returns the length of the array
80         if n[sample]<0:
81             temp=1
82         else:
83             temp=0
84         arr.append(temp) # Adds a single item to the existing list
85     plt.xlabel('n')
86     plt.ylabel('x(n)')
87     plt.title('Unit Step Discrete-Time Function for 0 ≤ n ≤ 9')
88     plt.step(t, arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
89     plt.show() # To view your plot
90
91 def impulse_d(): # Unit Impulse Continuous-Time Signal
92     t=np.arange(-10, 10) # Get evenly spaced values within a given interval
93
94 In [6]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')
95
96 MENU:
97 1. Sinusoidal Continuous-Time Signal.
98 2. Sinusoidal Discrete-Time Signal.
99 3. Exponential Continuous-Time Signal.
100 4. Exponential Discrete-Time Signal.
101 5. Unit Step Continuous-Time Signal.
102 6. Unit Step Discrete-Time Signal.
103 7. Unit Impulse Continuous-Time Signal.
104 8. Unit Impulse Discrete-Time Signal.
105 9. Ramp Function Continuous-Time Signal.
106 10. Ramp Function Discrete-Time Signal.
107 11. Exit
108 Enter the number corresponding to the menu to implement the choice: 6
109
110 In [7]:
```

Type here to search O I W F E Python console History LSP Python: ready conda (Python 3.8.8) Line 112, Col 14 UTF-8 CRLF RW Mem 85% 26°C ⚡ ENG 0054 28-09-2021



Unit Impulse Continuous-Time Signal

The screenshot shows the Spyder Python IDE interface. The code in the editor window is as follows:

```
t=np.linspace(-2, 10, 1000) # linspace() creates evenly spaced sequences
arr=[ ] # Null Array Declaration
for sample in range(len(t)): # Returns the length of the array
    if t[sample]==0:
        temp=1
    else:
        temp=0
    arr.append(temp) # Adds a single item to the existing list
plt.xlabel('t')
plt.ylabel('x(t)')
plt.title('Unit Step Continuous-Time Function')
plt.step(t,arr) # line forming a series of steps between data points
plt.show() # To view your plot

def step_d(): # Unit Step Discrete-Time Signal
    n=np.arange(-10, 10) # Get evenly spaced values within a given interval
    arr=[ ] # Null Array Declaration
    for sample in range(len(n)): # Returns the length of the array
        if n[sample]>0:
            temp=1
        else:
            temp=0
        arr.append(temp) # Adds a single item to the existing list
    plt.xlabel('n')
    plt.ylabel('x(n)')
    plt.title('Unit Step Discrete-Time Function for 0 <= n <= 9')
    plt.stem(n,arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
    plt.show() # To view your plot

def impulse_c(): # Unit Impulse Continuous-Time Signal
    t=np.arange(-2, 10) # Get evenly spaced values within a given interval
    arr=[ ] # Null Array Declaration
    for sample in range(len(t)): # Returns the length of the array
        if t[sample]==0:
            temp=1
        else:
            temp=0
        arr.append(temp) # Adds a single item to the existing list
    plt.xlabel('t')
    plt.ylabel('x(t)')
    plt.title('Unit Impulse Continuous-Time Function')
    plt.plot(t,arr) # line forming a series of steps between data points
    plt.show() # To view your plot

def impulse_d(): # Unit Impulse Discrete-Time Signal
    n=np.arange(-10, 10) # Get evenly spaced values within a given interval
```

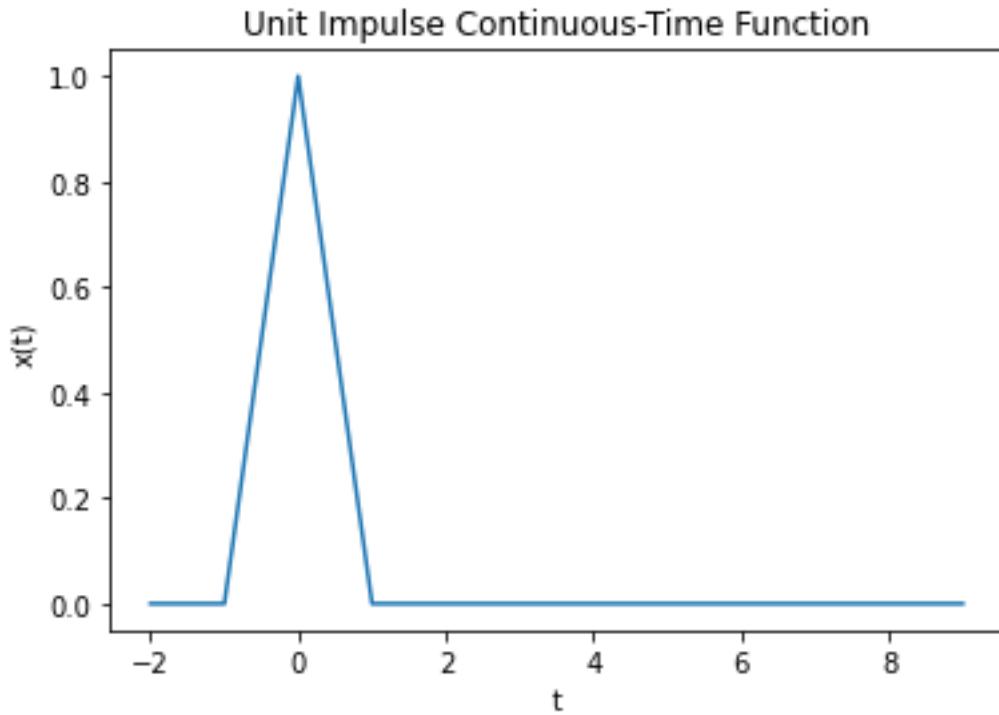
The plot window displays the title "Unit Impulse Continuous-Time Function". The x-axis is labeled "t" and ranges from -2 to 8. The y-axis is labeled "x(t)" and ranges from 0.0 to 1.0. A single blue vertical line segment is plotted at t=0, reaching a height of 1.0, representing the unit impulse function.

Console output:

```
In [7]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')
MENU:
1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit
Enter the number corresponding to the menu to implement the choice: 7
```

Python console:

```
LSP Python: ready
conda (Python 3.8.8) Line 112, Col 14 UTC-0 CRLF RW Mem 84%
26°C ⌂ 28-09-2021 0055 ENG
```



Unit Impulse Discrete-Time Signal

The screenshot shows the Spyder Python 3.8 IDE interface. The code editor displays a script named Experiment 2.py with the following content:

```
80 if n[sample]>=0:
81     temp=1
82     else:
83         temp=0
84         arr.append(temp) # Adds a single item to the existing list
85         plt.xlabel('n')
86         plt.ylabel('x(n)')
87         plt.title('Unit Step Discrete-Time Function for 0 ≤ n ≤ 9')
88         plt.stem(n,arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
89         plt.show() # To view your plot
90
91 def impulse_c(): # Unit Impulse Continuous-Time Signal
92     t=np.arange(-2, 10) # Get evenly spaced values within a given interval
93     arr=[] # Null Array Declaration
94     for sample in range(len(t)): # Returns the length of the array
95         if t[sample]==0:
96             temp=1
97             else:
98                 temp=0
99                 arr.append(temp) # Adds a single item to the existing list
100                plt.xlabel('t')
101                plt.ylabel('x(t)')
102                plt.title('Unit Impulse Continuous-Time Function')
103                plt.plot(t,arr) # Line forming a series of steps between data points
104                plt.show() # To view your plot
105
106 def impulse_d(): # Unit Impulse Discrete-Time Signal
107     n=np.arange(-2, 10) # Get evenly spaced values within a given interval
108     arr=[] # Null Array Declaration
109     for sample in range(len(n)): # returns the length of the array
110         if n[sample]==0:
111             temp=1
112             else:
113                 temp=0
114                 arr.append(temp) # Adds a single item to the existing list
115                 plt.xlabel('n')
116                 plt.ylabel('x(n)')
117                 plt.title('Unit Impulse Discrete-Time Function for 0 ≤ n ≤ 9')
118                 plt.stem(n,arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
119                 plt.show() # To view your plot
120
121 def ramp_c(): # Ramp Function Continuous-Time Signal
122     t=np.linspace(0, 10, 1000) # linspace() creates evenly spaced sequences
123     plt.plot(t,t)
124     plt.xlabel('t')
125     plt.show()
```

The plot area shows a discrete-time unit impulse signal $x(n)$ for $n \in [-2, 9]$. The signal is zero for all $n \neq 0$ and has a value of 1 at $n = 0$.

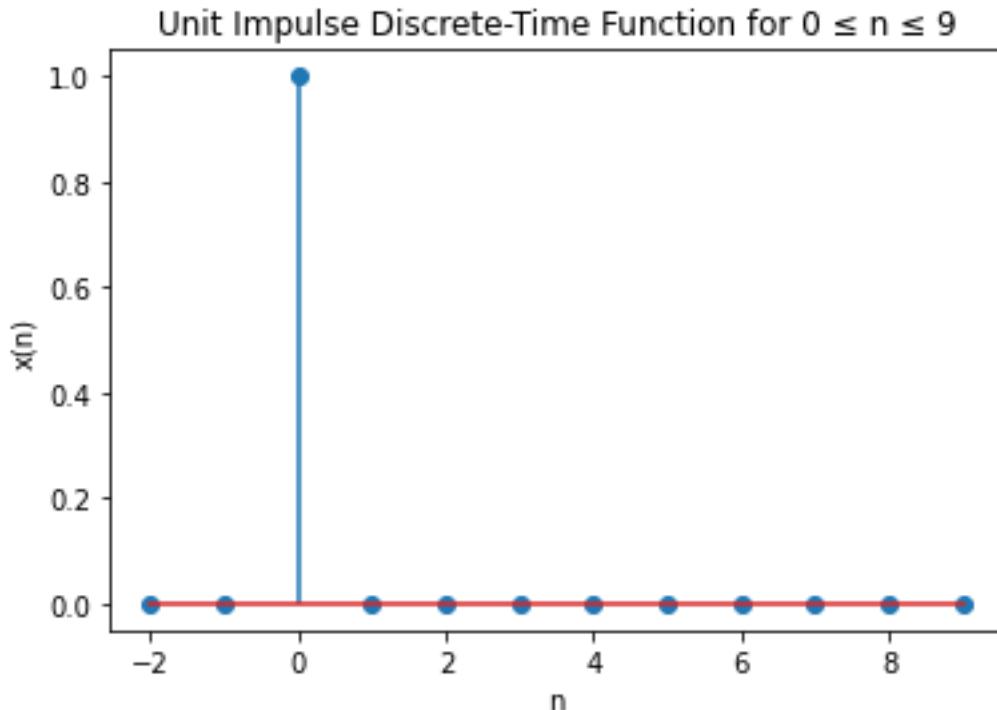
Console output:

```
In [8]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')
In [9]:
```

Menu options:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 8



Ramp Function Continuous-Time Signal

The screenshot shows the Spyder Python IDE interface. The code editor displays 'Experiment 2.py' with several signal processing functions: impulse_c, impulse_d, ramp_c, and ramp_d. The plot area shows a ramp function from t=0 to t=10 with a value from 0 to 10. A menu on the right lists signal types, and the console shows a selection prompt.

```
def impulse_c(): # Unit Impulse Continuous-Time Signal
    t=np.arange(-2, 10) # Get evenly spaced values within a given interval
    arr=[] # Empty array Declaration
    for sample in range(len(t)): # Returns the length of the array
        if t[sample]==0:
            temp=1
        else:
            temp=0
        arr.append(temp) # Adds a single item to the existing list
    plt.xlabel('t')
    plt.ylabel('x(t)')
    plt.title('Unit Impulse Continuous-Time Function')
    plt.plot(t,arr) # Line forming a series of steps between data points
    plt.show() # To view your plot

def impulse_d(): # Unit Impulse Discrete-Time Signal
    n=np.arange(-2, 10) # Get evenly spaced values within a given interval
    arr=[] # Empty array Declaration
    for sample in range(len(n)): # returns the length of the array
        if n[sample]==0:
            temp=1
        else:
            temp=0
        arr.append(temp) # Adds a single item to the existing list
    plt.xlabel('n')
    plt.ylabel('x(n)')
    plt.title('Unit Impulse Discrete-Time Function for 0 ≤ n ≤ 9')
    plt.stem(n,arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y and places a marker there.
    plt.show() # To view your plot

def ramp_c(): # Ramp Function Continuous-Time Signal
    t=np.linspace(0, 10, 1000) # linspace() creates evenly spaced sequences
    plt.xlabel('t')
    plt.ylabel('r(t)')
    plt.title('Ramp Continuous-Time Function')
    plt.ylim([0, 10]) # Set the x-limits of the current axes
    plt.show() # To view your plot

def ramp_d(): # Ramp Function Discrete-Time Signal
    n=np.arange(0, 10) # linspace() creates evenly spaced sequences
    plt.xlabel('n')
    plt.ylabel('r(n)')
    plt.show() # To view your plot
```

In [9]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')

Console 1/A

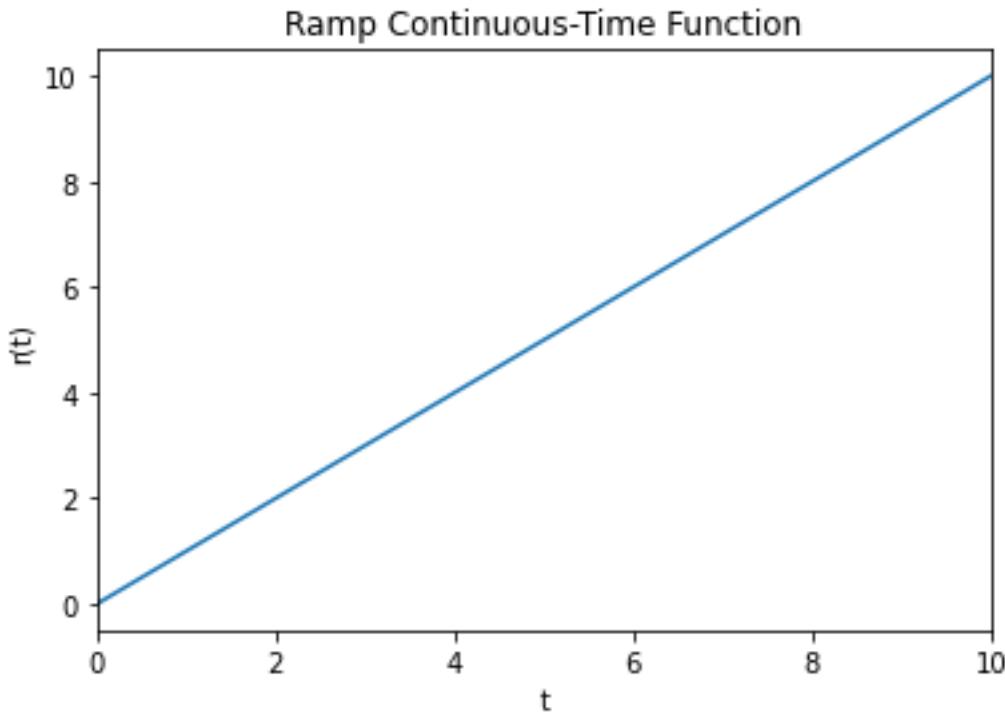
MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 9

In [10]:

Python console History



Ramp Function Discrete-Time Signal

The screenshot shows the Spyder Python IDE interface. The top bar has tabs for File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. Below the tabs, there are several icons for file operations. The main area has two panes: one for the code editor and one for the IPython console.

Code Editor:

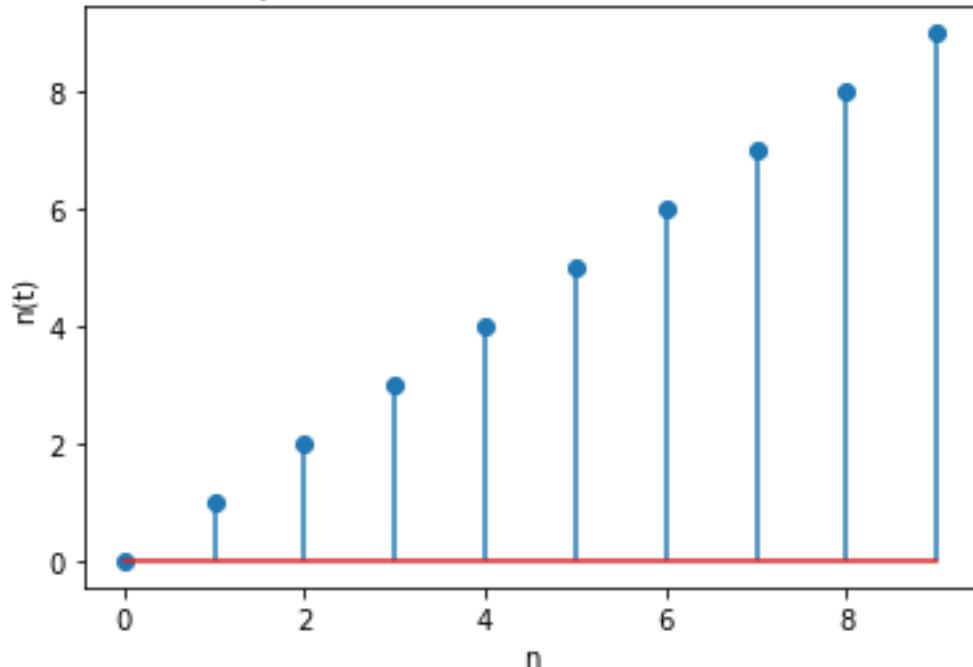
```
97     else:
98         temp=0
99         arr.append(temp) # Adds a single item to the existing list
100    plt.xlabel('n')
101    plt.title('Unit Impulse Continuous-Time Function')
102    plt.plot(t,arr) # line forming a series of steps between data points
103    plt.show() # To view your plot
104
105 def impulse_d(): # Unit Impulse Discrete-Time Signal
106     t=np.linspace(0, 10) # Creates evenly spaced values within a given interval
107     arr=[0]*len(t)
108     arr[0]=1 # Initial Value Declaration
109     for n in range(len(n)): # returns the length of the array
110         if n==sample==0:
111             temp=1
112         else:
113             temp=0
114             arr.append(temp) # Adds a single item to the existing list
115     plt.xlabel('n')
116     plt.title('Unit Impulse Discrete-Time Function for 0 ≤ n ≤ 9')
117     plt.stem(t,arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
118     plt.show() # To view your plot
119
120 def ramp_c(): # Ramp Function Continuous-Time Signal
121     t=np.linspace(0, 10, 1000) # linspace() creates evenly spaced sequences
122     plt.plot(t,t)
123     plt.xlabel('t')
124     plt.title('Ramp Continuous-Time Function')
125     plt.xlim(0,10) # Set the x-limits of the current axes
126     plt.show() # To view your plot
127
128 def ramp_d(): # Ramp Function Discrete-Time Signal
129     n=np.arange(0, 10) # linspace() creates evenly spaced sequences
130     arr=ramp_c()
131     plt.xlabel('n')
132     plt.title('Ramp Discrete-Time Function for 0 ≤ n ≤ 9')
133     plt.stem(n,n) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
134     plt.show() # To view your plot
135
136 # Main
137 while True: # This simulates a Do Loop
138     choice = input('
139     1. Sinusoidal Continuous-Time Signal
140     2. Sinusoidal Discrete-Time Signal
141     3. Exponential Continuous-Time Signal
142     4. Exponential Discrete-Time Signal
143     5. Unit Step Continuous-Time Signal
144     6. Unit Step Discrete-Time Signal
145     7. Unit Impulse Continuous-Time Signal
146     8. Unit Impulse Discrete-Time Signal
147     9. Ramp Function Continuous-Time Signal
148     10. Ramp Function Discrete-Time Signal.
149     11. Exit
150
151 Enter the number corresponding to the menu to implement the choice: 10
152
153 [In 11]: |
```

IPython Console:

```
In [10]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2/Experiment 2.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 2')
```

Plot: A stem plot titled "Ramp Discrete-Time Function for 0 ≤ n ≤ 9". The x-axis is labeled "n" and ranges from 0 to 9. The y-axis is labeled "ramp(n)" and ranges from 0 to 9. The plot shows discrete data points connected by vertical lines, starting at (0,0) and increasing linearly to (9,9).

Ramp Discrete-Time Function for $0 \leq n \leq 9$



Exit

The screenshot shows the Spyder Python 3.8 IDE interface. The code editor displays a script named 'Experiment 2.py' with the following content:

```
137 # Main
138 while True: # This simulates a Do Loop
139     choice = input(
140         "1. Sinusoidal Continuous-Time Signal.\n"
141         "2. Sinusoidal Discrete-Time Signal.\n"
142         "3. Exponential Continuous-Time Signal.\n"
143         "4. Exponential Discrete-Time Signal.\n"
144         "5. Unit Step Continuous-Time Signal.\n"
145         "6. Unit Step Discrete-Time Signal.\n"
146         "7. Unit Impulse Continuous-Time Signal.\n"
147         "8. Unit Impulse Discrete-Time Signal.\n"
148         "9. Ramp Function Continuous-Time Signal.\n"
149         "10. Ramp Function Discrete-Time Signal.\n"
150         "11. Exit\nEnter the number corresponding to the menu to implement the choice: ") # Menu Based Implementation
151
152 if choice == str(1): # str() returns the string version of the variable "choice"
153     sinusoidal_c() # Sinusoidal Continuous-Time Signal
154     break
155 elif choice == str(2):
156     sinusoidal_d() # Sinusoidal Discrete-Time Signal
157     break
158 elif choice == str(3):
159     exponential_c() # Exponential Continuous-Time Signal
160     break
161 elif choice == str(4):
162     exponential_d() # Exponential Discrete-Time Signal
163     break
164 elif choice == str(5):
165     step_c() # Unit Step Continuous-Time Signal
166     break
167 elif choice == str(6):
168     step_d() # Unit Step Discrete-Time Signal
169     break
170 elif choice == str(7):
171     impulse_c() # Unit Impulse Continuous-Time Signal
172     break
173 elif choice == str(8):
174     impulse_d() # Unit Impulse Discrete-Time Signal
175     break
176 elif choice == str(9):
177     ramp_c() # Ramp Function Continuous-Time Signal
178     break
179 elif choice == str(10):
180     ramp_d() # Ramp Function Discrete-Time Signal
181     break
182 elif choice == str(11):
183     break # Exit loop
184 else:
185     print("Error: Invalid Input! Please try again.\n")
```

The plot window shows a discrete-time ramp function for $0 \leq n \leq 9$. The signal starts at 0 for $n=0$ and increases linearly to 9 at $n=9$. The plot includes three smaller subplots showing different signal types.

The console window shows the menu options and the user's input '12' followed by an error message: 'Error: Invalid Input! Please try again.'

The status bar at the bottom right indicates: LSP Python: ready, conda (Python 3.8.8), Line 177, Col 1, UTF-8, CR/LF, RW, Mem 84%, 26°C, 01:00, 28-09-2021.

This screenshot is nearly identical to the one above, showing the same code in 'Experiment 2.py', the same plot of a discrete-time ramp function, and the same console output. The only difference is the timestamp in the status bar, which has changed from '01:00' to '01:01'.

Thank You!

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.
```

```
IPython 7.29.0 -- An enhanced Interactive Python.
```

```
Restarting kernel...
```

```
In [1]:      'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/  
Signal Processing Lab (19CCE281)/Assignments/Experiment 2/Experiment 2.py'      = 'E:/Plan  
B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab  
(19CCE281)/Assignments/Experiment 2'
```

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

```
Enter the number corresponding to the menu to implement the choice: 1
```

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

```
Enter the number corresponding to the menu to implement the choice: 2
```

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

```
Enter the number corresponding to the menu to implement the choice: 3
```

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 4

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 5

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 6

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 7

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 8

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 9

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 10

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.

11. Exit

Enter the number corresponding to the menu to implement the choice: 12
Error: Invalid Input! Please try again.

MENU:

1. Sinusoidal Continuous-Time Signal.
2. Sinusoidal Discrete-Time Signal.
3. Exponential Continuous-Time Signal.
4. Exponential Discrete-Time Signal.
5. Unit Step Continuous-Time Signal.
6. Unit Step Discrete-Time Signal.
7. Unit Impulse Continuous-Time Signal.
8. Unit Impulse Discrete-Time Signal.
9. Ramp Function Continuous-Time Signal.
10. Ramp Function Discrete-Time Signal.
11. Exit

Enter the number corresponding to the menu to implement the choice: 11

In [2]: